Practice Set

1. Write a program which will take an integer 'x' as input and then pass this variable as parameter to function void PrintAddress(int x); which will print the address of variable x.

1. Write a program which will take two characters as input from the user and call function void SwapChar(char a, char b) to swap the content of variable a with variable b.Print the swapped values in the main function.

1. Write a program which will take marks of three quizzes as input, then call a function void ScaleUp(float q1, float q2, float q3) to check if the marks of any of the quiz 'q' is less than the average of rest of two quizzes 'a'. If marks of any quiz is less than the average of the rest of two marks then add the difference between the average of the two quizzes and the quiz itself to q i.e., q = a - q, to scale up the quiz marks.

   For example, if q1 = 5, q2 = 8 and q3 = 9 then your function will change the marks of quiz 1 using the steps below:

   A = (q3 + q2)/2 => (8+9)/2=> 8.5
   Diff = A - q1 => 8.5 - 5 => 3.5
   Q1 = Q1 + Diff => 5 + 3.5 => 8.5

   Note that the marks are input from the user, so there might be three cases

   I. if q1 < q2 + q3 or
   Ii. if q2 < q1 + q3 or
   Iii. if q3 < q1 + q2

Practice Set for Pointers:

1. Write a program which will take a floating point number as input and call the function void breakFloating(float f, int *intPart, int *fracPart) which will separate the integer part and fractional-part of that floating point number and print those number (which are now stored in two integer variables) in main function

2. Repeat Question # 2 of the Practice Set for Reference to Operator using pointers.

3. Write a function that uses pointer to copy an array of size n. The datatype of the array should be double. You can take input from the user or you can use random variables

4. Write a function which will receive a pointer to array of floats of length n. The function will return the address of the maximum value in the array.

Problem Set for Dynamic Memory:

1. Write a program which will create a dynamically created 1D array of length n (given by the user) and store first n odd numbers in it. Try printing the size of the array using sizeof() function. Are you satisfied with the result of sizeof() function? If no, explain why?

1. Write a program which will create 1D array on heap of length n. Fill up the array by generating random values using **rand()%100** and then call a function to calculate mean, maximum and minimum of the numbers stored in the array. Print the values of the calculated variables in the main function.

Note: when you will use rand() function, you have to include cstdlib, using ***#include<cstdlilb>***

1. Write a program which will create a dynamically allocated 1D array of length n (given by the user) such that, 100 > n > 1000. Fill in the array with random variables using rand() %1000. Now print all the numbers, which appears **more than once** in the array.

For example, if your program generates the following 100 random numbers:
41 467 334 500 169 724 478 358 962 464 705 145 281 827 961 491 995 942 827 436 391 604 902 153 292 382 421 716 718 895 447 726 771 538 869 912 667 299 35 894 703 811 322 333 673 664 141 711 253 868 547 644 662 757 37 859 723 741 529 778 316 35 190 842 288 106 40 942 264 648 446 805 890 729 370 350 6 101 393 548 629 623 84 954 756 840 966 376 931 308 944 439 626 323 537 538 118 82 929 541

Then the output of your program should be:

8272 538 35

5. Write a function that will take a dynamic allocated array and size. Then it will delete the first element from a dynamic array. Then return the new pointer.

Hints:

```
int* addElement(int list[], int *siz, int value)
```

```
{
//Your code
}
```

6. The final grades of a student for his first 3 semesters at IUB are given below:

1st semester: A B A
2nd semester: B A A A
3rd semester: A B

Use ideas of dynamic memory where appropriate to encode the data given above in a two dimensional array. After printing the two dimensional array it would look like:

A	B	A	
B	A	A	A
A	B		

1. The array below is average daily temperature in Celsius of an imaginary city Audacity for 10 days in May.

1	19	20	12	30	11	15	25	35	13	-5

Your job is to categorize the temperature into 3 different category, and then combine them in two dimensional array.

<=10 degree celsius	-5	1			
>10 & <20 degree celsius	12	11	13	15	19
>=20 degree celsius	20	30	25	35	

Use dynamic memory where appropriate.

Problem Set for Struct:

1. Create a struct on Point like the one given below:

```
struct Point
{
   float x;
   float y;
};
```

Now dynamically create two variables of type Point. Take the values of x and y coordinates of both the points and print the values.

1. Create a struct of Patient which contains following information:
   Patient ID, Patient Name, Patient Contact Number
Add the data for at least 5 patients and allow user to search any of the item to get complete information of the patient.
For example, below is the code input/output expectations:

Sample Input / Output

```
Select the search item:
Press 1 to search by Patient ID
Press 2 to search by Patient Name
Press 3 to search by Patient's given contact number
3

Enter the Patient's registered Contact number:
01856332874
```

```
Patient's Record:
Patient ID: 707
Patient Name: Adil Ahsan Amin
Patient Contact Number: 01856332874
```

Note: You will have to dynamically create an array of length 5. Please read the lecture slides.

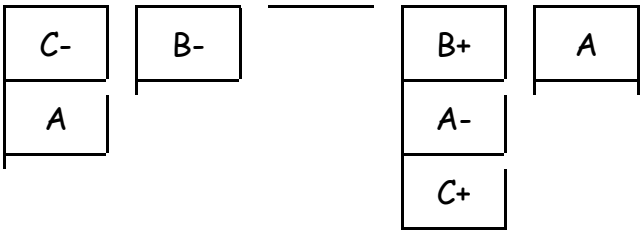
1. The final grades of a student in his 5 years of honours degree at IUB are given below:

1st Year: A B A C B+ C- A
2nd Year: B A A A B B-
3rd Year: A B A A A
4th Year: D A A B- B B+ A- C+
5th Year: B- B+ A A- A A

Use ideas of dynamic memory and struct where appropriate to encode the data given above. Your data structure should look theoretically like the one given below:

Num of Courses per year	Pointer to the array
7	1000
6	8000
5	6000
8	2000
6	3000

1000	8000	6000	2000	3000
A	B	A	D	B-
B	A	B	A	B+
A	A	A	A	A
C	A	A	B-	A-
B+	B	A	B	A

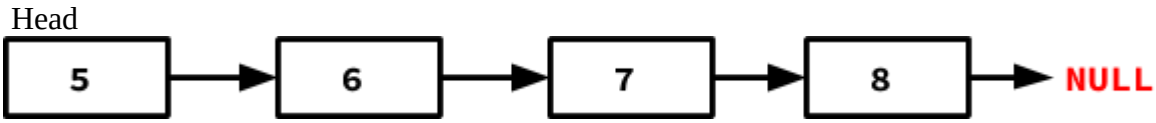


Hint: You can create an array of struct 'StudentGrades' which can hold two variables i.e., number of courses 'numCoursesr' taken by student in each year and a pointer 'p' to the base address of array of size equal to number of courses, which holds the grades earned by the student in different years.

Practice Set For Linked List:

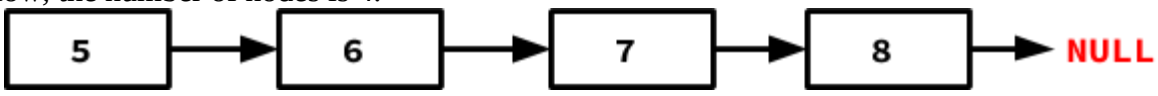
1. Given a node defined as :

```
struct node{
   int data;
   node *next;
       }
```



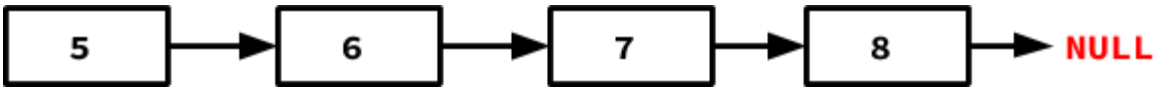
The nodes are connected using singly linked list. Current pointer defined as node* **Head** points to the element **5**. Find the average of the elements present from **5** till the end of the list.

2. Write code to print the number of nodes in a singly linked list. For example in the linked list below, the number of nodes is 4.



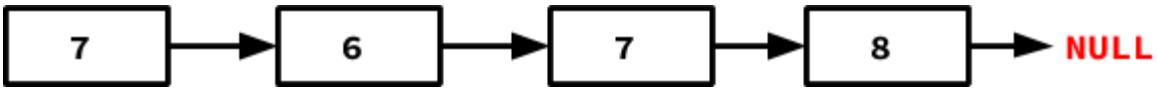
3. Write a function with the following signature:
   bool findElement(node *&list, int value)

The function will take a singly linked list and a value. It will return **true** if a value is present in the list, and will return **false** if a value is not found in the list.



For example if the value **6** is given in the function will return **true**, and if **-6** is given it will return **false**.

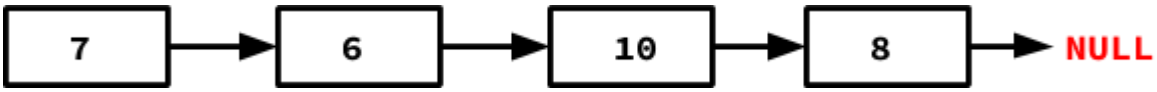
4. Write a function with the following template:
   int numOfOccurrences(node **list, int value)
The function will return the number of times a values occurred in a given list.



For example in the above list, if 7 is given as input as value, it will return 2.

5. Complete the following function
   int countEven(node *&list)
   {

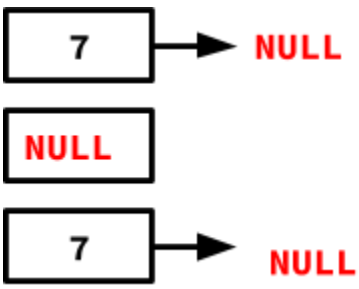
   }
The function will return the number of even numbers in a given list.

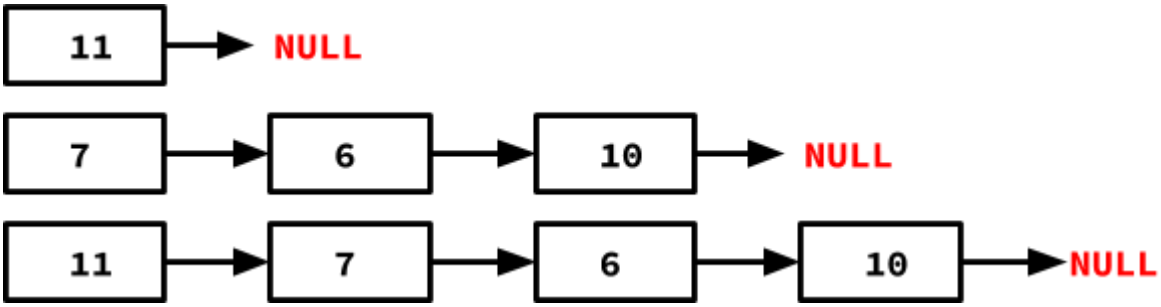


For example in the above list,the function will return **3**.

6. Complete the following functions for singly linked list:

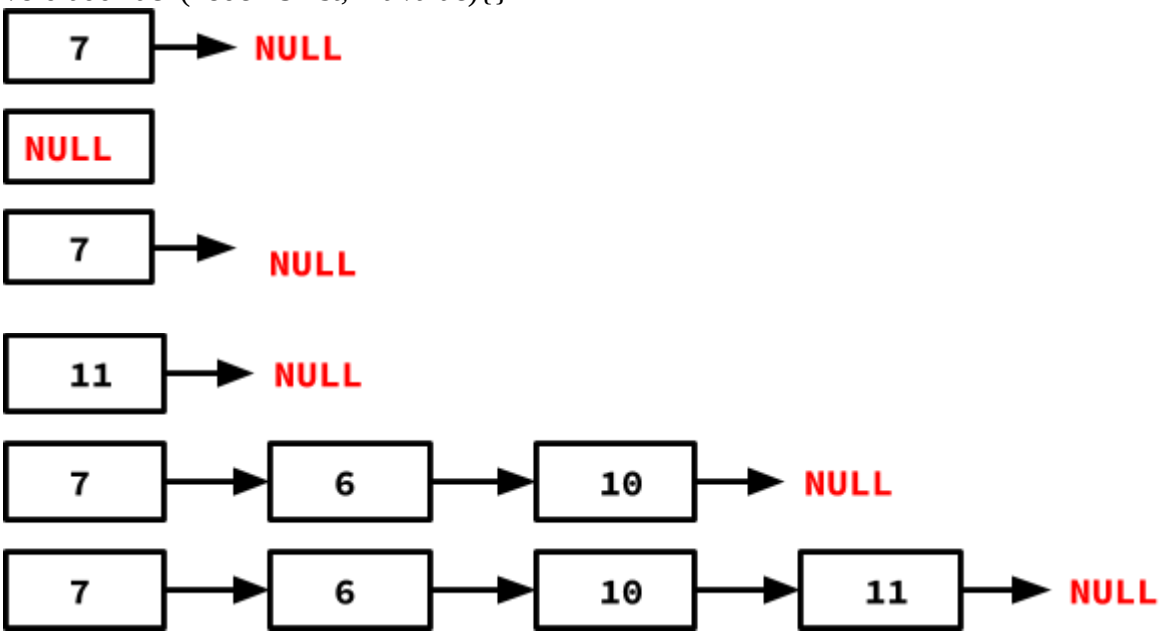
void addFront(node *&list, int value){}





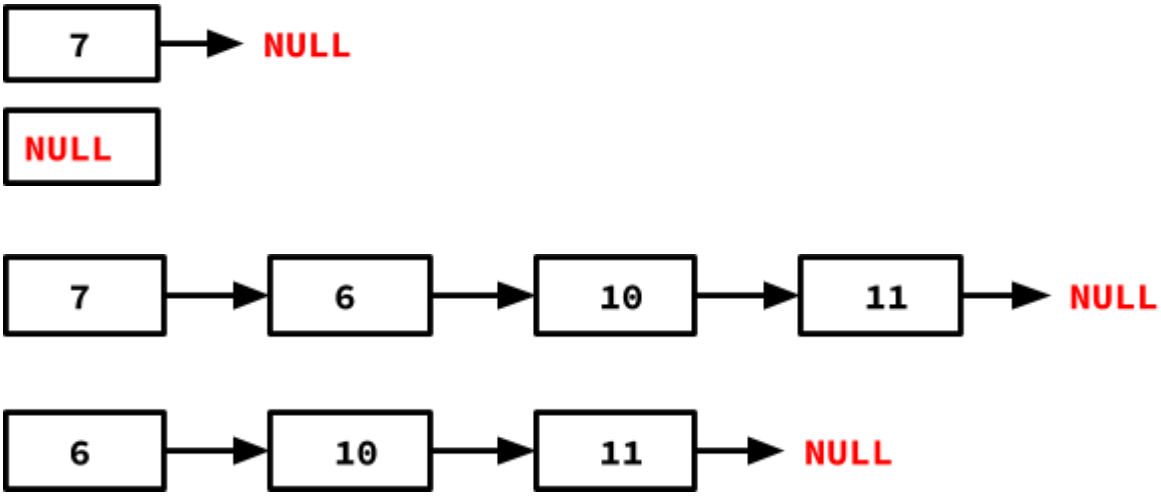
---

void addBack(node *&list, int value){}



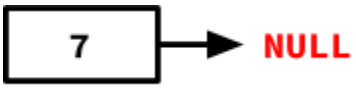
---

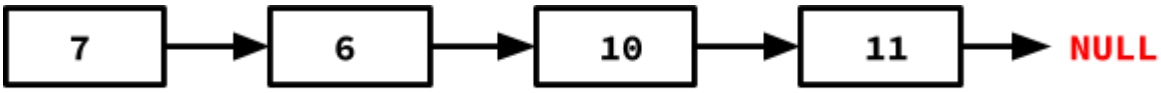
void removeFront(node *&list){}

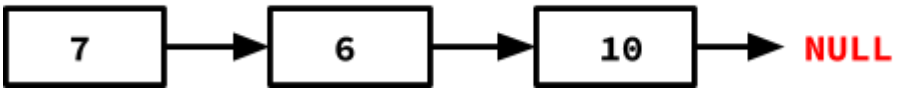


---

void removeBack(node *&list){}

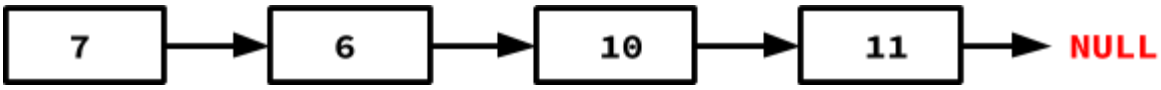






---

7. Write a function to write the value of the alternate nodes of a singly linked list.



The output of the program for the above linked list is 7, 10

8. Write a program to determine whether the list a is a subset of list b or not.

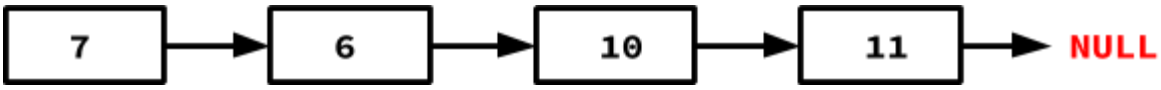


Figure: List A

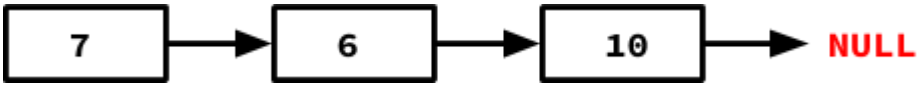


Figure: List B_1

Figure: List B_2

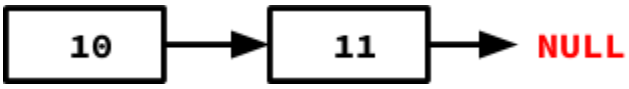


Figure: List B_3

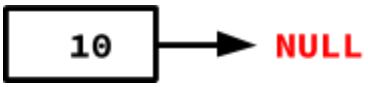


Figure: List B_4

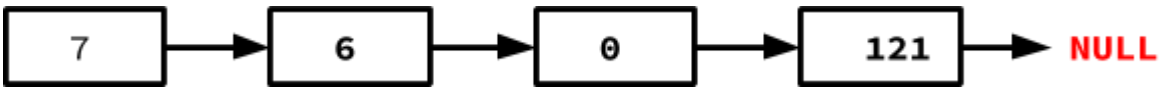


Figure: List B_5

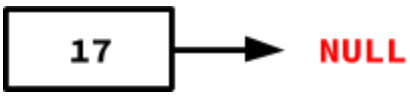


Figure: List B_6

In the above examples, list B_1, B_2, B_3 and B_4 are subset of list A. However, list B_5 and B_6 are not subset of list A.

9. You're given the pointer to the head nodes of two linked lists. Compare the data in the nodes of the linked lists to check if they are equal. The lists are equal only if they have the same number of nodes and corresponding nodes contain the same data.

Figure: List A

Figure: List B

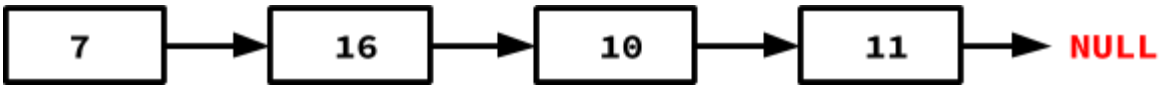


Figure: List C

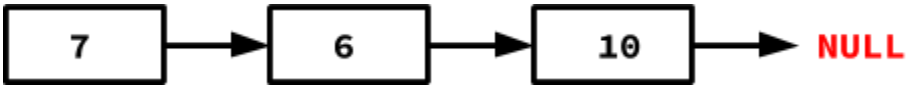


Figure: List D

For example a function call to equalNode(A,B) will return **true**, equalNode(A,A) will return **true**, equalNode(A,C) will return **false**, equalNode(A,D) will return **false**.

10. Write a function which takes two **sorted** list and returns another sorted list.

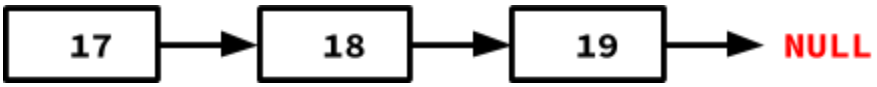


Fig: List A

Fig: List B

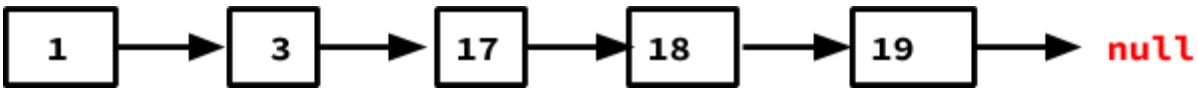


Fig: List C

For example in the above example if **sorted** List A and **sorted** List B are given as input in a function mergeList(A,B) , it will return List C which is **sorted**.

11. Write a function which takes two **unsorted** lists and returns a **sorted** list.

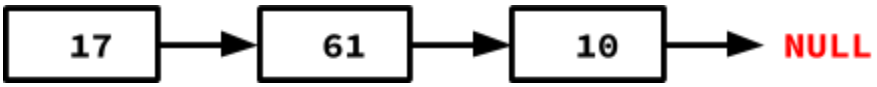


Fig: List A

Fig: List B

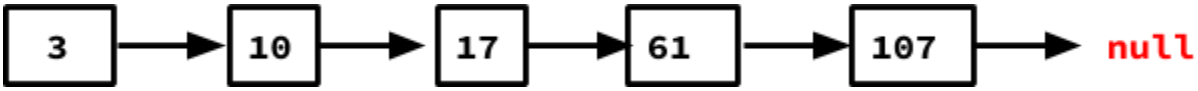


Fig: List C

For example in the above example if **unsorted** List A and **unsorted** List B are given as input in a function mergeList(A,B) , it will return a **sorted** List C.

12. Assume there is a singly linked list. HEAD is the pointer for head node. Write a function reverseList(node*&list) to reverse the linked list. It means, reverse the links. Then print the reversed list.

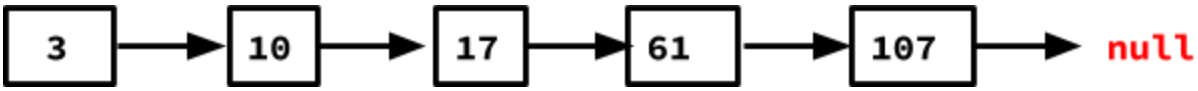


Fig A: List A

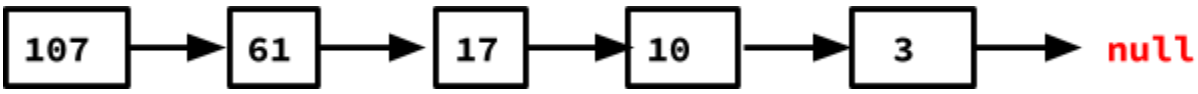


Fig B: List B

For example if 3 represents the head node in Fig A it will represent the tail in Fig B.

13. Remove the even number from the following linked list:

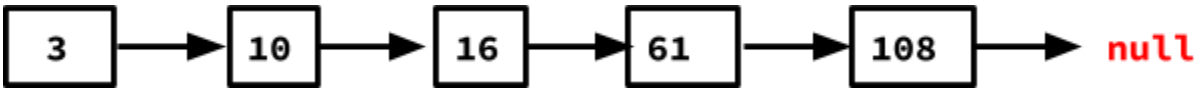


Fig A: List A

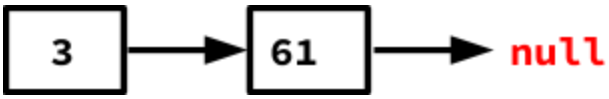


Fig B: List B

14. Remove all the duplicate ids except the first one from the list below:
The function signature for the given operation is
void deleteDuplicates(node*& list, int id){}

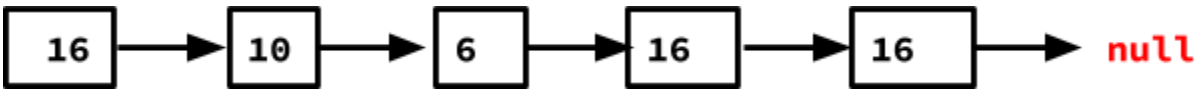


Fig A: List A

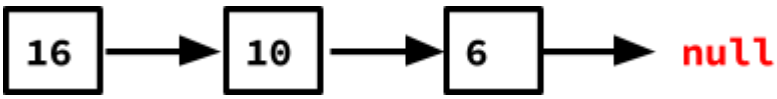


Fig B: List B

List A is the initial list and List B is the list after the duplicates except the first one is removed.

15. Assume, **HEAD** is the head-node of a singly linked list. Write a function addNodeMiddle() to insert a new node in the middle of the list.
Hints: a) If the size of the list is n, add the node after n/2.

16. Assume there is a singly linked list. **HEAD** is the pointer for head node. Write a function reverseList() to reverse the linked list. It means, reverse the links. Then print the reversed list.

Example:

List: 12 -> 15 -> 11 -> 8 -> 2
Shaded node is the HEAD.
Output:
2->8->11->15->12
Shaded node is New HEAD

17. Assume, HEAD is the head-node of a doubly linked list (**dll**). Write a function addNodeBeforeValue() to insert a new value in this **dll** just before a given value.

a) If the given value is found in the dll, insert the new value just before the given value.
b) If the given value is not found, don't add the node. Just print "Not found"

Stack

1.Suppose we wanted to implement a member function clear() which removed all the current entries from a stack. How would you implement such a function? Assume we have implemented functions of pop(), push(), isEmpty(). isFull() for a stack. We need to use those.

2.Suppose we have a stack and it is populated by integers and we know the TOP. Also we assume that push(), pop(), and isEmpty() are implemented for this stack. Write a program in a way that the largest value in that stack would be in the TOP position.

3. Take a string from the user and print it in the reverse order using **stack** data structure.

Input:

W A T E R M E L O N

Output:

N O L E M R E T A W

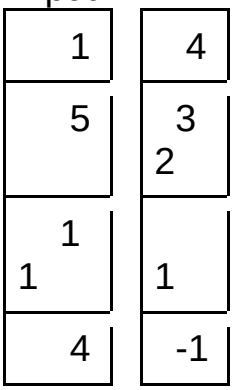
Hint: Use the property of stack to reverse the string

4. Given a string of brackets, determine whether each sequence of brackets is balanced. If a string is balanced, print YES on a new line; otherwise, print NO on a new line.

Input :	Output
()	Yes
{}	Yes
[]	Yes
{[]}	Yes
{([)]}	No
(()	No
{[(]}	No

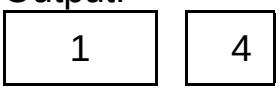
5. Given two stacks with integer values, balance the sum of integer values present in each of the stack by adding a new element in any of the stack. For example, if stack **A** contains 1, 5, 11 and 4 and stack **B** contains 4 ,32,-1 and 1 then by adding 15 in stack **A** will balance the sum of values of both the stacks.

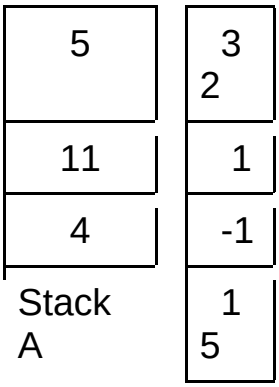
Input:



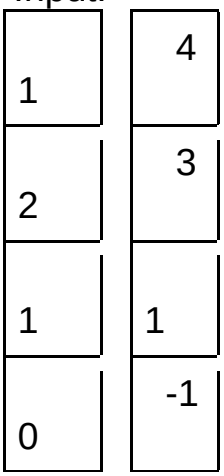
Stack A Stack B

Output:

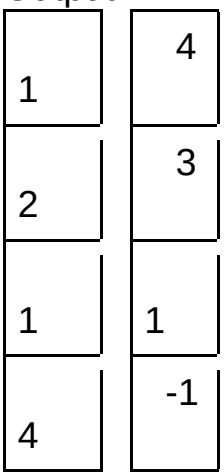




Input:



Output:



6. Program a simple text editor using Stack. The user will input a complete sentence. You have to put this string into a character stack. When the user selects the option to undo, a word from the stack will

be deleted. By selecting undo again, another word from the stack will be removed and so on.

7. Suppose we wanted to implement a member function clear() which removed all the current entries from a stack. How would you implement such a function? Assume we have implemented functions of pop(), push(), isEmpty(). isFull() for a stack. We need to use those.
8. We have a **Stack**. And it is populated by integers. Also we assume that push(), pop(), and isEmpty() are implemented for this **Stack**. Write a program in a way that the largest value in that **Stack** would be in the TOP position.

Hints: You should use a helpingStack and its functions helpingStackPush(), helpingStackPop(), and helpingStackisEmpty(). Assume: functions helpingStackPush(), helpingStackPop(), and helpingStackisEmpty() are already implemented. You need to use them properly.

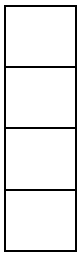
After your Program: Stack looks like below:

99
23
53
56
19
44
22
44

TOP

23
53
56
19
44
22
99
44

TOP



Stack **helperStack**

9. Given a string of parenthesis. Write a function **checkBalance()** to check whether the string has balanced parenthesis or not. Hints: Use a stack. Read one character, check it against the TOP character in stack. If no match, push to stack.
   Assume: stack functions are already implemented. Just use functions push(), pop(), isEmpty()

   Example:

	ut
{[()()]()}"	nced
	balanced

10. We have a stack. The stack is maintaining the ascending order always. Now, you would need to insert a new value into this stack using a **pushStack(**in value**)** function. Write the **pushStack()** function so that the function always maintains the ascending order in the stack. Assume normal push() and pop() functions are already there for the stack.

Queue

7. Reverse the first 'k' elements of a queue, push it back into the queue. For example, if the queue has members 10, 20, 30, 40 , 50 , 60 , 70, 80, 90 and first 5 numbers are asked to be reversed then the result will be 60, 70, 80, 90, 50, 40, 30, 20, 10.

Input:

10	**20**	**30**	**40**	**50**	60	70	80	90

Output:

60	70	80	90	**50**	**40**	**30**	**20**	**10**

8. You have two volunteers who are collecting donations from the members of a community. Suppose each of the members has a member id and they are lined up in ascending order. To keep the number of people balanced, volunteer A is collecting donations from members whose ids are divisible by 2 and volunteer B is collecting donations from members whose ids are not divisible by 2. Write a program which uses two queues, such that if the even number member id is given, it will assign that member to volunteer A, otherwise, the member will go to volunteer B.

Input:

2 4 7 10 1 9 11

Output:

2	4	10		

Queue A

7	1	9	11	

Queue B

9. Given a number n, write a function that generates and prints all binary numbers with decimal values from 1 to n. You need to use a **queue** for that .

Input:

5

Output:

0
01
10
11
100
101

Input:

3

Output:

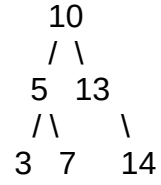
0
01
10
11

Input:
1
Output:
0
01

10. Assume there is a queue implemented using singly linked list. And we know FRONT node and REAR node pointer. Write enqueue() and dequeue() for this queue.

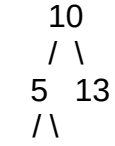
Binary Search Tree Problem Sets

1.Write a function that will print the level of a binary search tree. (Hint use a queue)

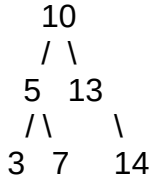


The output of the function will be:
10
5 13
3 7 14

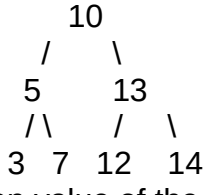
2.Write a function that will print the largest element in each level of a binary search tree.



The output of the given tree when the function is applied:
10
13
7
3. Write a function to find the median value of a given binary search tree.

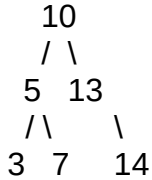


The median value of the above binary tree is 8.5

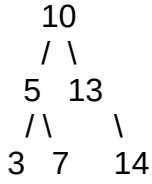


The median value of the above binary tree is 10

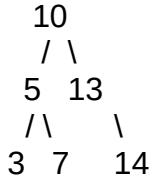
4. Check whether the leaf nodes of a BST is even or odd.



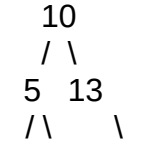
The output of the BST above is [odd, odd, even].
5. Find the predecessor and successor of 10 in the following BST.



The predecessor of 10 is 7 and the successor is 13.
6. Find the total number of nodes in a BST.



The total number of nodes in the above tree is 6.
7. Write a function code to print the leaf nodes of a BST.



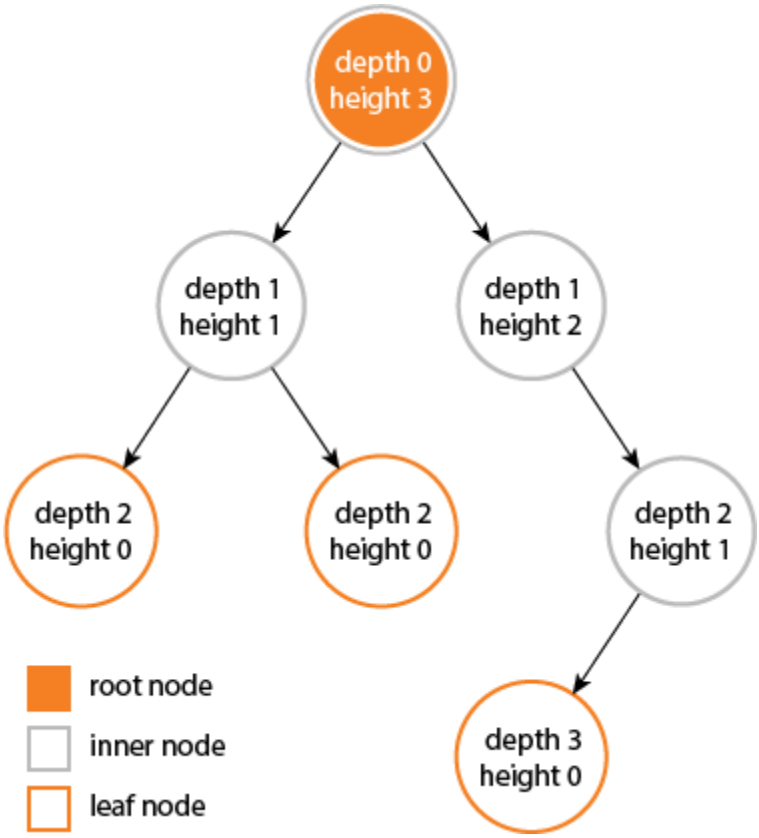
3 7 14

The output of the function is [3,7,14].

8. Find height and depth of a BST.

```
        10
       /  \
      5   13
     / \    \
    3   7    14
```

Depth and height of a BST is defined below.



(Link:https://stackoverflow.com/questions/2603692/what-is-the-difference-between-tree-depth-and-height)

```
        10
       /  \
      5   13
     / \    \
    3   7    14
```

Fig: A

/

4

Fig: B

9.Write a function to remove odd numbers from a BST. Keep in mind about freeing memory. Removing odd numbers from Figure A will produce a BST like that in Figure B.

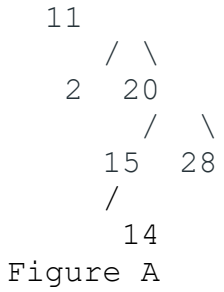


Figure A

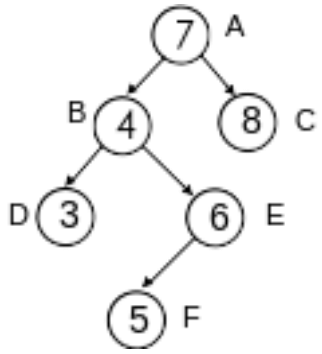
Figure B

10.

Given a node defined as:

```
struct Node{
   int value;
   int numOfDescendants;
   Node* left;
   Node* right;
};
```

Traverse a BST and store in the numOfDescendants variable, the number of descendants that each of the nodes in the BST contains.



For example in the above figure:

A has 5 descendants

B has 3 descendants

C has 0 descendant
D has 0 descendant
E has 1 descendant
F has 0 descendant

Graph Problems

1. Given a graph as shown in Figure 1, find the height of the starting node for a random run.
The function signature is:
int **getDepth**(Graph *g, int start)

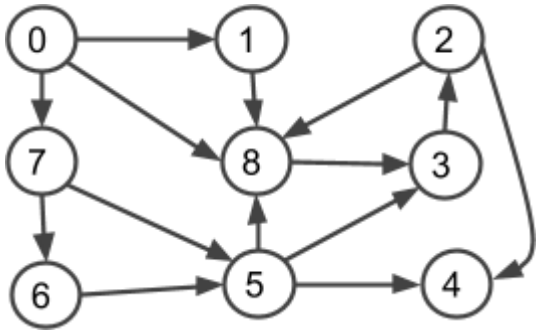


Figure: 1

2. Write code to check whether a node **n1** in a graph **G** is reachable from node **n2**. If yes, return true, else return false. For example, in figure 1, node 4 is not reachable from node 5; node 3 is not reachable from node 4 , but node 4 is accessible from node 3.
The function signature is:
bool **isReachable**(Graph *g, int source, int destination)

3. Print each level of the graph in Figure 1 using modified BFS. One possible output of your code would give the following result:

```
0
1 8 7
5 6 3
4 2
```

The function signature is:
void **printLevel**(Graph *g)

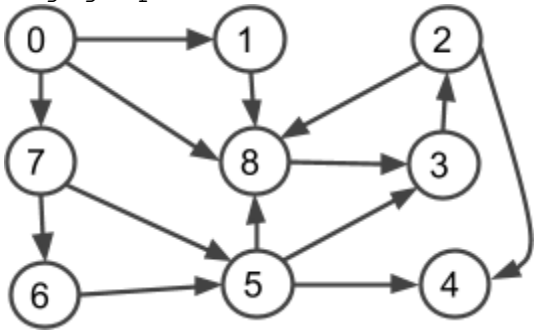
4. Given a source node **src,** find all the path that can be traversed to reach node **dest.**
For example if src is **0** and dest is **8** - the paths will be:
0 1 **8**
0 **8**
0 7 5 **8**
0 7 6 5 **8**

If src is **0** and dest is **4** - the paths will be:

0 1 8 3 2 **4**
0 8 3 2 **4**
0 7 5 8 3 2 **4**
0 7 5 3 2 **4**
0 7 6 5 3 2 **4**
0 7 6 5 **4**
0 7 5 **4**
0 7 6 5 8 3 2 **4**

The function signature is:
void **printPath**(Graph *g,int src, int dest)

5. Write code to find the least distance of src to the dest for the following graph.



The function signature is:
int **getLeastDistance**(Graph *g,int src, int dest)

6.

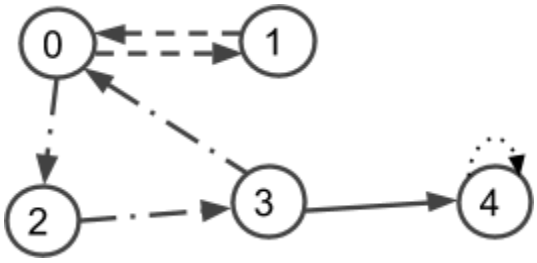


Figure: 3

Write code to count the number of cycles in a directed graph.
For example in figure 3, the number of cycles is 3.
0->2->3->0
4->4
0->1->0

The function signature is:
int **countCycles**(Graph *g)

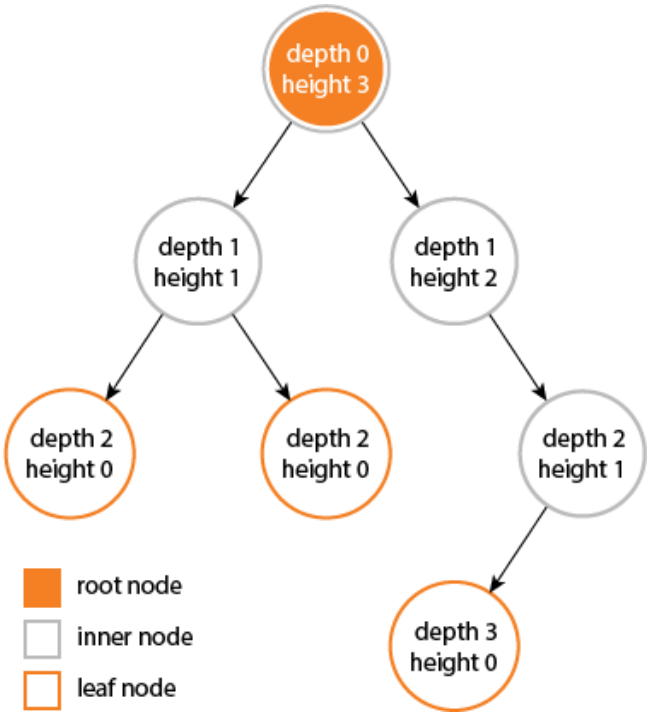


Figure: 2

Solve the following problems using Recursion

1. Write a program to find the sum of first n numbers.
2. Given a string, check whether the string is a palindrome or not .
3. Given a number, check whether it is palindrome or not.
4. Given two integers, find and print the GCD (Greatest Common Divisor) of them.
5. Reverse a string using Recursion
6. Change the display function of a linked list to print all the value of nodes recursively
7. Delete k^{th} Node in a linked list using Recursion

Solve the following problems using Pointer Referencing

1. Take a set of numbers as input from the user and print it in the fashion of Queue using two Stacks. There will be only one Push(int) and Pop() function which will take top of Stack 1 and Stack 2 as argument and perform the operation on respective Stack accordingly.

1. Write a program which will take input from the user until the user enters -1. If the number entered by user is:

a. Prime; then insert it in the linked list of prime numbers named "prime" in sorted manner

b. Perfect; then insert it in the linked list of perfect numbers named "perfect" in sorted manner

c. Neither, then insert it in the linked list named "dust_bin" in sorted manner

At the end of user input, print all three linked lists. Note that, you have only one insert Function which will take head and number as argument and place it in the respective linked list

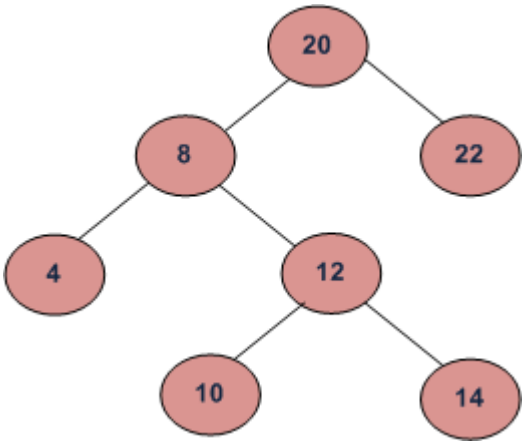
BST and Min heap

1. Check whether two BSTs are identical or not. Write a function bool isDuplicate(Node* root1, Node* root2) that will check whther the BST with root1 is exactly identical to the other BST with root2.
2. Write a function int findHeight(Node* root) to find he height of the BST with root.

3. Write a function int findSuccessor(Node*root, Node* node) that will return the value of the successor of the Node **node**.
4. Write a function int findPredecessor(Node*root, Node* node) that will return the value of the predecessor of the Node **node**.
5. Write a function int findLCA(Node*root, Node* nodeA, Node* nodeB) that will return the value of the lowest common ancestor of the given nodeA and nodeB.

Detail Hints: https://www.geeksforgeeks.org/lowest-common-ancestor-in-a-binary-search-tree/

Example:




```
LCA of 10 and 14 is 12
LCA of 14 and 8 is 8
LCA of 10 and 22 is 20
```

6. Write findMin(Node* root) and findMax(Node* root) iteratively and recursively.
7. Write a Function bool isBST(Node* root) to check whether the tree with **root** is a valid BST or not.
8. Write a function int calculatesum(Node* root) to calculate sum of all nodes of the BST with root. Please write the function iteratively and recursively.
9. Write a function int countNodes(Node* root) to calculate number of all nodes of the BST with root. Please write the function iteratively and recursively.

10. Write insert function for BST using iteration and also with recursive way.
11. Search a value in a BST. Iteratively and recursively
12. Print in order of BST values, Iteratively and recursively 6. Print Preorder of a BST
13. Print a BST with sorted order, Iteratively and recursively
14. Build a BST from an array. Write a program.
15. Delete a key in a BST, Iteratively and recursively
16. An array A of size 10 is given.

12	34	11	44	11	34	21	45	15	3

A

Find the fourth largest value in this array using MinHeap. Also find the fourth largest value using MaxHeap

17. Write a recursive function bool isPelinDrome(string str) recursively. This function will check whether a given string **str is a palindrome or not.**
18. Build max heap, min heap from a given array
19. Find k the smallest value of an array using Min heap. Assume insertKey(int value), extractMin() are already implemented.
20. Build min heap from a array. Then use it to print the array in sorted order..
21. Develop a priority queue that would stores student id and cgpa. Here is the priority rules.
    a. Less id has highest pritity
    b. If same id, higher cgpa has higher priority
22. Develop a priority from an array. Priority rules are as follow
    a. The array element that has smaller value has higher value
    b. If value same, then the value that comes first (smaller index) has the higher value