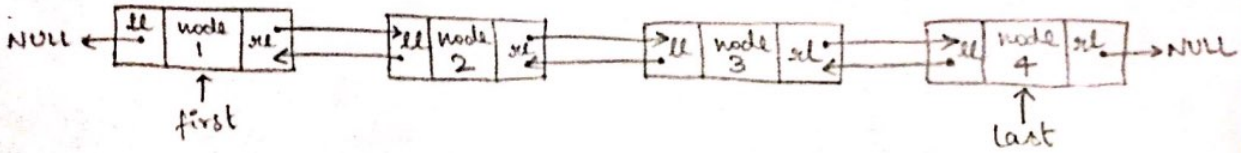
1. What is doubly linked list ? How it is advantageous over singly linked list ?

Doubly linked list is a linked data structure that consists of a set of sequentially linked records called nodes. Traversal of such a list is possible from both ends as each node in a doubly linked list contains two fields, called links, where the left link refers to the previous node and the right link refers to the next node respectively.



Advantages of doubly linked list over singly linked list -

* A doubly linked list can be traversed in both the directions either from first or from last, but a singly linked list can be traversed only in one direction from the first.
* A node deletion is more simpler in a doubly linked list as we have pointers for both previous and next node, but in a singly linked list as the pointer to the previous node is not available is becomes tougher.
* Immediate access to the first and the last node is available in a doubly linked list but in a singly linked list only immediate access to the first node is available.
* Doubly linked list enables the easy implementation of all operations than the singly linked list as every node has easy access to its previous and next nodes.

2. Give algorithm for Search (X) and InsertFirst (Y) in a doubly linked list.

(i) Search (X) :

Step 1 : START
Step 2 : Create a temporary node - "temp".
Initialise temp to point to first.
Step 3 : Check if the list is empty.
If so display "Empty list" and goto step 7.
Step 4 : Repeate step 5 to step 6 until temp is not equal to NULL.
Step 5 : Check if data in temp equals X
If so display "found" and goto step 8
Step 6 : Set temp to next node i.e. to its ri
Step 7 : Print "Element not found".
Step 8 : STOP.

(ii) Insert First (Y) :

Step 1 : START
Step 2 : Create a temporary node - "temp".
Initialise temp data with Y.
Step 3 : Check if the list is empty.
If so make first and last to point to temp.
goto step 7.
Step 4 : Set right link of temp to first.
Step 5 : Set left link of first to temp.
Step 6 : Set first to temp.
Step 7 : STOP.

3. Write a function to add two polynomials represented by two singly linked list, A and B and returns the new polynomial C.

```
#include<bits/stdc++.h>
using namespace std;
class Poly;
class Node
{
    int c,e;
    Node *next;
  public:
    Node(int c1, int e1)
    {
        c=c1; e=e1;
        next = NULL;
    }
    friend class Poly;
};
class Poly
{
    Node *first;
  public:
    Poly()
    { first = NULL; }
    void InsertLast(int c, int e)
    {
        Node *temp;
        temp = new Node(c,e);
        if(first==NULL)
            first = temp;
        else
        {
            Node *navi;
            navi = first;
```

```
            while (navi->next)
            {
                navi = navi->next;
            }
            navi->next = temp;
        }
    }
void display()
{
    Node *temp;
    temp = first;
    while (temp)
    {
        cout << temp->c;
        if (temp->e > 1)
            cout << "x^" << temp->e;
        else if (temp->e == 1)
            cout << "x";
        temp = temp->next;
        if (temp)
            cout << "+";
    }
    cout << "\n";
}
void create()
{
    int ans = 1;
    do
    {
        cout << "Enter terem to insert \n";
        int c,e;
        cin>>c>>e;
        Insertlast(c,e);
        cout << "want to continue 1 or 0 ? \n";
        cin >> ans;
    } while (ans == 1);
}
```

```
Poly add (Poly p1)
{
    Poly p;
    Node *t1 = first;
    Node *t2 = p1.first;
    while (t1 && t2)
    {
        if (t1->e > t2->e)
        {
            p.Insertlast (t1->c, t1->e);
            t1 = t1->next;
        }
        else if (t1->e < t2->e)
        {
            p.Insertlast (t2->c, t2->e);
            t2 = t2->next;
        }
        else
        {
            p.Insertlast (t1->c + t2->c, t1->e);
            t1 = t1->next;
            t2 = t2->next;
        }
    }
    while (t1)
    {
        p.Insertlast (t1->c, t1->e);
        t1 = t1->next;
    }
    while (t2)
    {
        p.Insertlast (t2->c, t2->e);
        t2 = t2->next;
    }
    return p;
}
};
```

2

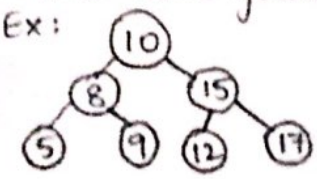
```
int main()
{
    Poly A, B, C;
    A.create();
    B.create();
    C = A.add(B);
    A.display();
    B.display();
    C.display();
}
```

4. Define the following terms and illustrate with an example:

(i) Binary search tree -

Is a special case of binary tree in which for any node, say A, elements in the left subtree are less than info of A and elements in right subtree are greater than or equal to info of A.

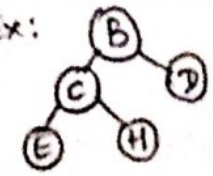
Ex:



(ii) Strictly Binary tree -

If the out degree of every node in a binary tree is either 0 or 2 (1 not allowed), then the tree is strictly binary tree.

Ex:



→ level 0
→ level 1
→ level 2 (max level)

(iii) Height of a Binary tree -

Height of a tree is one more than the maximum level in the tree. Ex: Height of above two trees = 3.

(iv) Height Balanced tree -

5. Explain with algorithm and example, the following tree traversal techniques :

(i) Pre order -

It is processed recursively as -
* Process the root
* Traverse the left subtree
* Traverse the right subtree

Step 1 : START
Step 2 : check if root is equal to NULL.
If so goto step 6.
Step 3 : Print data in root.
Step 4 : Repeat the algorithm with left ree.
Step 5 : Repeat the algorithm with righ. o tree.
Step 6 : STOP

(ii) Inorder -

It can be processed recursively as -
* Traverse the left subtree
* Process the root
* Traverse the right subtree

Step 1 : START
Step 2 : Check if root is equal to NULL.
If so goto step 6.
Step 3 : Repeat the algorithm with left-subtree.
Step 4 : Print data in root.
Step 5 : Repeat the algorithm with right subtree.
Step 6 : STOP.

(iii) Post order -

It can be processed recursively as -
* Traverse the left subtree
* Traverse the right subtree
* Process the root

Step 1 : START

Step 2 : Check if root is equal to NULL;
If so goto step 6.

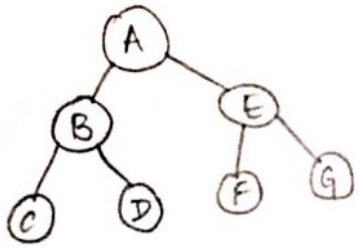
Step 3 : Repeat the algorithm with left-subtree.

Step 4 : Repeat the algorithm with right-subtree.

Step 5 : Print data in root.

Step 6 : STOP.

Ex :



Preorder - A B C D E F G

Inorder - C B D A F E G

Post order - C D B F G E A

6. Given two singly linked lists A and B, representing 2 sorted lists, give the algorithm to create a new linked list C, by merging these two lists so that the resultant list should be a sorted list.

Step 1 : START

Step 2 : Create temporary reference t1 and t2 to traverse through list A and B respectively.

Step 3 : Create a temporary node - "temp" to be inserted to C.

Step 4 : Repeate step 5 to 6 until A or B reach NULL.

Step 5 : check if data in node t1 of A is lesser than data in node t2 of B
If so copy data in t1 to temp and insert temp to C and move t1 to next

7.

Step 6 : Since condition in Step 5 fails, copy data in t2 to temp and insert temp to C and move t2 to next node.

Step 7 : Check if A has reached NULL, Else copy the remaining list in A to C.

Step 8 : Check if B has reached NULL, Else copy the remaining list in B to C.

Step 9 : Print list C.

Step 10 : STOP.

7. Given two polynomials, write an algorithm to perform polynomial addition.

Step 1 : START

Step 2 : Input 2 polynomials A and B.

Step 3 : Scan through A and B till any one of them reach NULL and repeate step 4 to 6.

Step 4 : check if the power of the term in A is greater than the power of the term in B. If so add the term in A to C.

Step 5 : check if the power of the term in A is lesser than the power of the term in B. If so add the term in B to C.

Step 6 : Since condition in 4 & 5 fails the power are equal, so add the co-efficients of the two terms in A and B and add the term to C.

Step 7 : Check for remaining terms in A and add them to C.

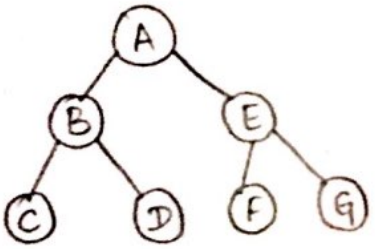
Step 8 : Check for remaining terms in B and add them to C.

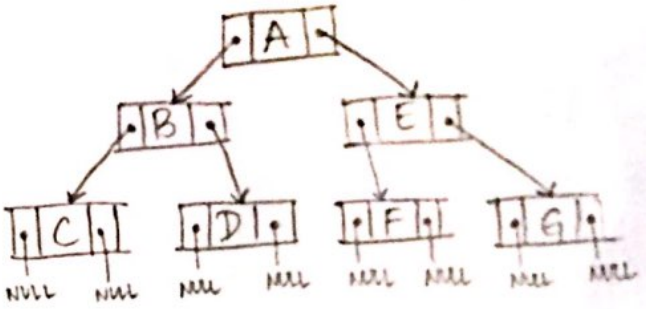
Step 9 : Print C.

Step 10 : STOP.

2

8. Illustrate the linked list representation of binary trees by taking an example.






```
#include <bits/stdc++.h>
using namespace std;
class Btree;
class Node
{
        int data;
        Node *lc, *rc;
    public:
        Node(int x)
        {   data = x;
            lc = rc = NULL;
        }
        friend class Btree;
};
class Btree
{
        Node *root;
    public:
        Btree()
        {   root = NULL; }
        void createBT()
        {
            cout << "Enter Root \n";
            int x;
            cin >> x;
            root = create(x);
        }
```

```
Node * create (int x)
{
    if (x == -1)
        return NULL;
    Node * temp = new Node (x);
    int y;
    cout << " Enter left child of " << x << "\n";
    cin >> y;
    temp->lc = create(y);
    cout << " Enter right child of " << x << "\n";
    cin >> y;
    temp->rc = create(y);
    return temp;
}
};
```

4. Write a C++ program to insert the record of a st. (name, rollno, age) in a doubly linked list at a node specified by position.

```
#include <bits/stdc++.h>
using namespace std;
class DLStudent;
class Node
{
    int rollno, age;
    char name[100];
    Node * llink, * rlink;
  public:
    Node (char nm[100], int r, int a)
    {
        rollno = r;
        age = a;
        strcpy (name, nm);
        llink = rlink = NULL;
    }
    friend class DLStudent;
```

```
class DLStudent
{
    Node * first, * last;
public :
    DLStudent ()
    {
        first = last = NULL;
    }
    int IsEmpty()
    {
        if ( first == NULL && last == NULL)
            return 1;
        return 0;
    }
    void Insertfront (char nm[100], int r, int a)
    {
        Node * temp;
        temp = new Node (nm, r, a);
        if ( IsEmpty() )
            first = last = temp;
        else
        {
            temp -> rllink = first;
            first -> llink = temp;
            first = temp;
        }
    }
    void Insertlast (char nm[100], int r, int a)
    {
        Node * temp;
        temp = new Node ( nm, r, a);
        if (IsEmpty())
            first = last = temp;
        else
        {
```

```
            temp->llink = last;
            last->rlink = temp;
            last = temp;
        }
    }
void create()
{
    int ans=1;
    do
    {
        cout<<"Enter data\n";
        int r,a,x;
        char nm[100];
        cin>>nm>>r>>a;
        cout<<"specify position 1. Insert front
                2. Insert last \n";
        cin>>x;
        if(x==1)
            Insertfront(nm,r,a);
        else if(x==2)
            Insertlast(nm,r,a);
        cout<<"want to continue? 1 or 0 \n";
        cin>>ans;
    } while(ans==1);
  }
};
int main()
{
    DLStudent ob;
    ob.create();
}
```

12. write a C++ program to implement a linked stack.

```
#include <bits/stdc++.h>
using namespace std;
class Linklist;
class Node
{
    int data;
    Node *next;
  public:
    Node (int x)
    {
        data = x;
        next = NULL;
    }
    friend class linklist;
};
class linklist
{
    Node *top;
  public:
    linklist()
    {  top = NULL;  }
    void push (int x)
    {
        Node *temp;
        temp = new Node (x);
        temp -> next = top;
        top = temp;
    }
    void Traverse()
    {
        Node *temp;
        temp = top;
```

```
        while (temp)
        {
            cout << temp->data << "\n";
            temp = temp->next;
        }
    }
int pop()
{
    if (top == NULL)
    {
        cout << "Empty list";
        return -99;
    }
    Node * temp;
    temp = top;
    int x = top->data;
    top = top->next;
    delete temp;
    return x;
}
};
```

11. What do you mean by a doubly linked list? Give the structure of a doubly linked list in C++. Also write an algorithm to delete a node before a given node in a doubly linked list.

Doubly linked list - Refer Q1.

```
#include <bits/stdc++.h>
using namespace std;
class DLinklist;
class Node
{
    int data;
    Node *llink, *rlink;
```

```
public:
    Node(int x)
    {
        data = x;
        llink = rlink = NULL;
    }
    friend class Dlinklist;
};
class Dlinklist
{
    Node *first, *last;
public:
    Dlinklist()
    {
        first = last = NULL;
    }
    int IsEmpty();
    void Insertfront(int x);
    void Insertlast(int x);
    void Traverse();
    void create();
}
```

-Algorithm to delete a node before a given node.

Step 1 : START

Step 2 :

12. Implement stack operations using doubly link list.

```
#include <bits/stdc++.h>
using namespace std;
class Dlinklist;
class Node
{
    int data;
    Node *rlink, *llink;
  public:
    Node(int x)
    {
        data = x;
        rlink = llink = NULL;
    }
    friend class Dlinklist;
};
class Dlinklist
{
    Node * top;
  public:
    Dlinklist()
    {
        top = NULL;
    }
```

```
void push (int x)
{
    Node * temp;
    temp = new Node (x);
    if (top = = NULL)
        top = temp;
    else
    {
        temp -> rlink = top;
        top -> llink = temp;
        top = temp;
    }
}
void Traverse ()
{
    Node * temp = top;
    while (temp)
    {
        cout << temp -> data << "\n";
        temp = temp -> rlink;
    }
}
int pop()
{
    if (top = = NULL)
    {
        cout << "Empty list";
        return -99;
    }
    Node * temp;
    temp = top;
    int x = top -> data;
    top = top -> rlink;
    delete temp;
    return x;
}
};
```

13. Define complete binary tree and binary search tree with an example for each.

(i) Complete Binary tree -

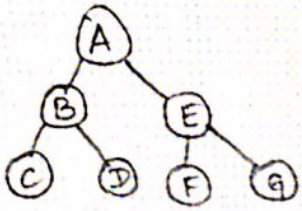
Is a strictly binary tree in which the number of nodes at any level 'i' is pow(2,i).

No. of leaf nodes = 2^d

No. of non-leaf nodes = $2^d - 1$

All leaf nodes are at level d. (d - depth of the tree).

Ex:



(ii) Binary search tree -

Refer Q.4 (i).

14. Write algorithm for depth first search. Explain with an example.

Depth first Search -

Preorder.
Inorder. Refer Q.5.
Postorder.

15. Write a program to find intersection of two doubly linked list.

```
#include <bits/stdc++.h>
using namespace std;
class Dlinklist;
class Node
{
    int data;
    Node *llink, *rlink;
  public:
    Node (int x)
    {
        data = x;
        rlink = llink = NULL;
    }
    friend class Dlinklist;
};
```

```
class Dlinklist
{
    Node *first, *last;
  public:
    Dlinklist()
    {
        first = last = NULL;
    }
    int IsEmpty()
    {
        if( first == NULL && last == NULL )
            return 1;
        return 0;
    }
    void Insertlast(int x)
    {
        Node *temp;
        temp = new Node(x);
        if (IsEmpty())
            first = last = temp;
        else
        {
            temp->llink = last;
            last->rlink = temp;
            last = temp;
        }
    }
    int Search(int x)
    {
        Node *temp = first;
        if (IsEmpty())
            return -99;
        while (temp)
        {   if (temp->data == x)
                return 1;
            temp = temp->rlink;
```

```
Dlinklist Intersection (Dlinklist ls2)
{
    Dlinklist res;
    Node *temp = ls2.first;
    while (temp)
    {
        if (search(temp->data))
            res.Insertlast(temp->data);
        temp = temp->rlink;
    }
    return res;
}
void create();
};
```

16. Write the following functions for doubly linked list.

(i) Insert_rear(list, ele)
(ii) Insert_front(list, ele)
(iii) Deleteval(list, ele)
(iv) Display(list)

(i)
```
void Insert_rear(Dllst ls, int ele)
{
    Node *temp;
    temp = new Node(ele);
    if (ls.first == NULL && ls.last == NULL)
        ls.first = ls.last = temp;
    else
    {
        temp->llink = ls.last;
        ls.last->rlink = temp;
        ls.last = temp;
    }
}
```

(ii) void Insert_front (Dlist ls, int ele)

```
{
    Node *temp;
    temp = new Node (ele);
    if ( ls.first == NULL && ls.last == NULL)
        ls.first = ls.last = temp;
    else
    {
        temp->rlink = ls.first;
        ls.first->llink = temp;
        ls.first = temp;
    }
}
```

(iii) void Deleteval (Dlist ls, int ele)

```
{
    if (ls.first == NULL && ls.last == NULL)
    {
        cout << "Empty list \n"; return;
    }
    Node *temp = ls.first;
    while (temp)
    {
        if (temp->data == x)
            break;
        temp = temp->rlink;
    }
    if (temp == NULL)
    {
        cout << "Data not found \n"; return;
    }
    Node *prev, *next;
    prev = temp->llink;
    next = temp->rlink;
```

```
if (prev && next)
{
    prev->rlink = next;
    next->llink = prev;
}
else if (prev == NULL)
{
    next->llink = NULL;
    ls.first = next;
}
else if (next == NULL)
{
    prev->rlink = NULL;
    ls.last = prev;
}
delete temp;
}
```

(iv)

```
void Display (Dlist ls)
{
    Node *temp = ls.first;
    while (temp)
    {
        cout << temp->data << "\n";
        temp = temp->rlink;
    }
}
```

17.