C++

10 Seconds
Harshit.

PAGE NO. 1
DATE 10/02/25

# Data Structures

## Singly linked list:

- create class with two fields - data and link as below

```
#include<iostream>
using namespace std;
class Node{
    public:
        int data;
        Node* next;
        Node(int value){
            data = value;
            next = NULL;
        }
};
int main(){
    Node *head;
    head = new Node(5);
    cout << head -> data;
}
```

Function to create Nodes // Insert at start

```
Node* create(Node* head, int value){
    if(head==NULL)
        head= new Node(value);
    else{
        Node* temp = new Node(value);
        temp -> next = head;
        head=temp;
    }
    return head;
}
```

The main function for the above

```
int main(){ Node* head =NULL; for(int i=1;i<=5;i++)
head = create(head,i); print(head);
```

10/02/25

The print function used to display a SLL:

```
void print(Node* head){
    while (head != NULL){
        out << head -> data << "\t";
        head = head -> next;
    }
}
```

The function to add a new node at the end/tail.

```
Node* insert_at_end(Node* head, int value){
    if (head == NULL)
        head = new Node(value);
    else {
        Node* temp1 = new Node(value);
        Node* temp = head;
        while (temp -> next != NULL)
            temp = temp -> next;
        temp -> next = temp1
    }
    return head;
}
```

To add a node after certain position in the singly linked list.

```
Node* insert_after(Node* head; int value, int pos){
    Node* temp1 = new Node(value);
    Node* temp = head;
    for(int i=0; i< pos-1; i++)
        temp= temp -> next;
    temp1 -> next = temp -> next;
    temp -> next = temp1;
    return head;
}
```

To delete the node at the start of the linked list.

PAGE NO 1
DATE 10/02/25

```
Node* delete_at_start (Node* head){
        if (head == NULL)
            cout << "No head found";
        NODE* temp = head;
        head = temp -> next
        return head;
}
```

To delete the node at the end of the linked list.

```
void delete_at_end (Node* head){
        if (head == NULL)
            cout << "No head found";
        NODE* temp = head;
        while (temp -> next != NULL && temp -> next -> next != NUL
            temp = temp -> next;
        temp -> next = NULL;
}
```

To delete any node at any position present in the linked list, without deleting either the nodes before or after it.

```
Node* delete_at_position(Node* head, int pos){
        if(head == NULL){
            cout << "List is empty.\n";
            return head;
        }
        if (pos == 0){
            NODE* temp = head;
            head = head -> next;
            delete temp;
            return head;
        }
        Node* temp = head;
```

10/02/25

```
for(int i=0; temp != NULL && i< pos-1 ;i++){
        temp = temp -> next;
}
if (temp==NULL || temp -> next ==NULL){
        cout << "Invalid position.\n";
        return head;
}
Node* temp1 = temp -> next;
temp -> next = temp1 -> next;
delete temp1;
return head;
}

// Create a new node
// Insert at start
// Insert at end
// Insert after
// Delete at start
// Delete at end
// Delete at any position.

The main function calling all the functions.
int main(){
    Node* head = NULL;
    for (int i=1; i<=5; i++)
        head = create(head,i);
    head = insert_at_start(head,6);
    head= insert_at_end(head,0);
    head = insert_after(head,10,0);
    delete_at_end(head);
    head = delete_at_start(head);
    head = delete_at_position(head,3);
    print(head);
} // 10 5 4 2 1
```

## Data Structures.

1. Write a program to print the middle elements in the linked list. If even, return highest.

```
Node* Reverse (Node* head){ //Program to reverse a SLL
    if (head == NULL)
        cout<< "No head exists";
    else if (head -> next == NULL)
        cout<< "Only one Node exists";
    else {
        Node* prev = NULL;
        Node* cur = head;
        Node* fut = head;
        while( cur != NULL) {
            fut = cur -> next;
            cur -> next = prev;
            prev = cur;
            cur = fut;
        }
        head=prev;
    }
    return head;
}
// find the middle element, if even return high.
void find_themidle (Node* & head){
    Node* head1 = head;
    Node* head2 = head;
    int count = 0;
    while (head1 != NULL) head1=head1->next, count++;
    for(int i=0; i<count/2; i++)
        head2 = head2->next;
    cout << head2 -> data << endl;
}
```

2. Write the code to implement a Doubly Linked list in C++ with functions for insertion, deletion, reversal, finding the middle node & displaying the list.

```
= #include<iostream>
using namespace std;

class Node{
public:
    int data;
    Node* next;
    Node* before;
    Node(int value){
        data= value;
        before = NULL;
        next = NULL;
    }
};
Node* create(Node* head, int value){
    if(head== NULL) head = new Node(value);
    else{
        Node *temp = new Node(value);
        temp -> next = head;
        head -> before = temp;
        head = temp;
    }
    return head;
}
void print(Node* head){
    while(head != NULL){
        cout<< head -> data << "\t";
        head = head -> next;
    }
    cout<<endl;
}
```

```
Node* insert_at_start (Node* head, int value){
    Node* temp = new Node(value);
    if (head != NULL){
        temp->next = head;
        head->before = temp;
    }
    head = temp;
    return head;
}

Node* insert_at_end (Node* head, int value){
    Node* temp1 = new Node(value);
    if (head == NULL) return temp1;
    Node* temp = head;
    while (temp->next != NULL) temp = temp->next;
    temp->next = temp1;
    temp1->before = temp;
    return head;
}

Node* insert_after(Node* head, int value, int pos){
    Node* temp1 = new Node(value);
    Node* temp = head;
    for (int i=0; temp != NULL && i < pos-1; i++)
        temp = temp->next;
    if (temp == NULL){
        cout << "Position out of range\n";
        return head;
    }
    temp1->next = temp->next;
    if (temp->next != NULL) temp->next->before = temp1;
    temp->next = temp1;
    temp1->before = temp;
    return head;
}
```

```
void delete_at_end(Node* &head){
    if(head==NULL){
        cout<<"No head found\n";
        return;
    }
    if(head->next==NULL){
        delete head;
        head=NULL;
        return;
    }
    Node* temp=head;
    while(temp->next->next!=NULL) temp=temp->next;
    delete temp->next;
    temp->next=NULL;
}
Node* delete_at_start(Node* head){
    if(head==NULL){
        cout<<"No head found\n";
        return NULL;
    }
    Node* temp=head;
    head=temp->next;
    if(head) head->before=NULL;
    delete temp;
    return head;
}
Node* delete_at_position(Node* head, int pos){
    if(head==NULL){
        cout<<"List is empty\n";
        return head;
    }
    if(pos==0) return delete_at_start(head);
    Node *temp=head;
```

```
for (int i=0; temp!=NULL && i<pos-1 ; i++)
        temp = temp->next;
if (temp==NULL || temp->next==NULL){
        cout<<"Invalid position.\n";
        return head;
}
Node* tempI = temp->next;
temp->next = tempI->next;
if (tempI->next != NULL) tempI->next->before=temp
delete tempI;
return head;
}
Node* Reverse(Node* head){
    if (head==NULL || head->next==NULL) return head;
    Node* prev=NULL;
    Node* cur = head;
    while(cur!=NULL){
        prev = cur->before;
        cur->before = cur->next;
        cur->next = prev
        cur = cur->before;
    }
    if (prev!=NULL) head=prev->before;
    return head;
}
void find_the_middle(Node* head){
    if(head==NULL){
        cout<<"List is empty.\n";
        return;
    }
    Node* slow = head;
    Node* fast = head;
    while(fast!
```

```
while( fast != NULL && fast -> next != NULL){
        slow = slow -> next;
        fast = fast -> next -> next;
    }
    cout<< " Middle element: " <<slow-> data<<endl;
}
int main(){
    Node* head = NULL;
    for(int i=1; i<=5; i++)
        head = create (head, i);
    print (head);
    head = insert_at_start (head, 6);
    print (head);
    head = insert_at_end (head, 0);
    print (head);
    head = insert_after (head, 10, 4);
    print (head);
    delete_at_end (head);
    print (head);
    head = delete_at_start (head);
    print (head);
    head = delete_at_position (head, 3);
    print (head);
    head = Reverse (head);
    print (head);
    find_the_middle (head);
}
```

10 Seconds
Harshit

PAGE NO. 1
DATE 12/02/25

# Data Structures

1. Check if the brackets in a string are in the correct format

```
#include<iostream>
#include<vector>
#include<stack>
using namespace std;

int main(){
    string a;
    cout<<"Enter a string of brackets: ";
    cin>>a;
    stack<char> st;
    for(int i=0; i<a.size(); i++){
        if(a[i]==')'){
            if(st.empty() || st.top()!='('){
                cout<<0;
                return -1;
            }
            st.pop();
        }else if(a[i]=='('){
            st.push(a[i]);
        }
    }
    if(st.empty()){
        cout<<1;
    }else{
        cout<<0;
    }
    return 0;
}
```

12/02/25

2. Given a list of strings, remove all consecutive duplicate strings and return the resulting list.

```
#include<iostream>
using namespace std;
#include <vector>
#include<stack>
int main(){
    vector<string> a = {"ab", "bc", "cd", "cd", "bc", "cd"};
    stack<string> st, temp;
    for(int i=0; i< a.size(); i++){
        if (st.empty()) st.push(a[i]);
        else if (st.top() != a[i]) st.push(a[i]);
        else st.pop();
    }
    while(!st.empty()) temp.push(st.top()), st.pop();
    while (!temp.empty())
        st.push(temp.top()), temp.pop(), cout<< st.top();
}
```

3. Remove all elements from the stack if the sign of the current element if different from the sign of the top element.

```
#include<iostream>
using namespace std;
#include<vector>
#include<stack>
int main(){
    vector<int> a= {5,4,2,-3,6,-4,-5,7};
    stack<int> st, temp;
    for(int i=0; i<a.size(); i++){
    if((st.empty())||(st.top()>0 && a[i]>0)||(st.top()<0&&a[i]<0))
        st.push(a[i]);
    else st.pop();
}
```

```
while (!st.empty())
        temp.push(st.top()), st.pop();
while(!temp.empty())
        st.push(temp.top()), temp.pop(), cout<<st.top();
}
```

4. Add the size of the stack to the top of the stack.

=
```
#include <iostream>
using namespace std;
#include<string>
#include<stack>
int main(){
    stack<int> st,temp;
    for(int i=0; i<=5; i++) st.push(i);
    while(!st.empty()){
        cout<<st.top()<<"\t";
        temp.push(st.top());
        st.pop();
    }
    cout<<endl<<endl;
    int n=6;
    st.push(n);
    for(int i=0; i<=5; i++) st.push(temp.top()), temp.pop();
    while(!st.empty()){
        cout<<st.top()<<"\t";
        st.pop();
    }
    cout<<endl;
}
```

12/02/25

5. Reverse a string using stack.

```
#include<iostream>
using namespace std;
#include<vector>
#include<stack>
int main(){
    string s;
    cin>>s;
    stack<char> st;
    for(int i=0;i<s.size();i++) st.push(s[i]);
    s="";
    while(!st.empty()){
        s+=st.top();
        st.pop();
    }
    cout<<endl<<s<<endl;
}
```

6. Add and remove elements from the end of the vector.

```
#include<iostream>
using namespace std;
#include<vector>
int main(){
    vector<int> a={1,2,3,4,5};
    for(int i=0;i<a.size();i++) cout<<a[i]<<"\t";
    cout<<endl;
    a.push_back(6);
    for(int i=0;i<a.size();i++) cout<<a[i]<<"\t";
    cout<<endl;
    a.pop_back();
    for(int i=0;i<a.size();i++) cout<<a[i]<<"\t";
    cout<<endl;
}
```

10 Seconds
Harshit

PAGE NO. 3
DATE 14/02/25

# Data Structures

1. A party has been organised on cruise. The party is organised for a limited time(T). The number of guests entering (E[i]) and leaving (L[i]) the party at every hour is represented as elements of the array. The task is to find the maximum number of guests present on the cruise at any given instance within T hours.

```
#include<iostream>
using namespace std;
int party(int T, int E[], int L[]){
    int temp[T], count=0;
    temp[0]=0;
    for(int i=0; i<T; i++)
        temp[i]= temp[i-1]+E[i]-L[i];
    for(int i=1; i<T; i++)
        if(count<temp[i])
            count = temp[i];
    return count;
}
int main(){
    int E[]={7,8,5,1,3};
    int L[]={1,2,1,3,4};
    int T=5;
    cout<< party(T,E,L);
```

2. N-base notation is a system for writing numbers that use only n different symbols, This symbols are the first n symbols from the given notation list (including the symbol to 0) Decimal to n base notation are (0:0, 1:1, 2:2, 3:3, 4:4, 5:5, 6:6, 7:7, 8:8, 9:9, 10:A, 11:B and upto 35:Z)

```
#include<iostream>
using namespace std;
```

```
string DectorNBase(int n, int num){
    string r;
    while(num>0){
        int x=num%n;
        if (n<10) r+=char('0'+x);
        else r+=char('A'+x-10);
        num=num/n;
    }
    return r;
}
int main(){
    ~~int~~ cout<< DectorNBase(12,718);
}
```

3. You are given a function, Int MaxExponents(int a, int b);
You have to find & return the number between 'a' & 'b' which has the maximum exponent of 2.

```
= #include<iostream>
using namespace std;
int maxExponents(int a, int b){
    int max=0, numtemp=0;
    for(int i=a; i<=b; i++){
        int count=0, n=i;
        while (n%2==0){
            n=n/2;
            count++;
        }
        if (numtemp<count){
            max=i;
            numtemp=count;
        }
    return max;
}
```

```
int main(){
    cout << maxExponents(3,9);
}
```

4. An automobile company manufactures both a two wheeler & a four wheeler (TW & FW). A company manager wants to make the production to make the production of both types of vehicle according to the given data below:
1st data, Total number of vehicle = V
2nd data, Total number of ~~vehicle~~ wheels = W

```
= #include<iostream>
using namespace std;
int main(){
    int TW=0, FW=0, V=200, W=540;
    FW = (W-2*V)/2;
    TW = V-FW;
    cout<<"TW="<<TW<<" FW="<<FW;
}
```

5. The function accepts an ~~integer~~ array 'arr', its length and two integer variables 'num' & 'diff', Implement this function to find and return the elements of 'arr' having an absolute difference of 'arr' having an absolute difference of less than or equal to 'diff' with 'num'.

```
= #include<iostream>
#include<cmath>
using namespace std;
int findCount(int arr[], int length, int num, int diff){
    int count=0;
    for(int i=0; i<length; i++){
        if(abs(arr[i]-num)<diff)
            count++;
    return count;
}
```

14/02/25

```
int main(){
    int a[] = {12, 3, 14, 56, 77, 13};
    cout << findCount(a, 6, 13, 2);
}
```