50 DSA Programs with Full Code

Domains: Arrays, Strings, Linked Lists, Trees, Graphs (C++)

1. Array - Find Maximum

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n; cin>>n;
    int a[n];
    for(int i=0;i<n;i++) cin>>a[i];
    cout<<*max_element(a,a+n);
}
```

2. Array - Reverse

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n; cin>>n;
    int a[n];
    for(int i=0;i<n;i++) cin>>a[i];
    reverse(a,a+n);
    for(int i:a) cout<<i<<" ";
}
```

3. Array - Kadane

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n; cin>>n;
    int a[n];
    for(int i=0;i<n;i++) cin>>a[i];
    int cur=a[0], mx=a[0];
    for(int i=1;i<n;i++){
        cur=max(a[i],cur+a[i]);
        mx=max(mx,cur);
    }
    cout<<mx;
}
```

4. String - Palindrome

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    string s; cin>>s;
    string t=s;
    reverse(t.begin(),t.end());
    cout<<(s==t?"YES":"NO");
}
```

5. String - Anagram

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    string a,b; cin>>a>>b;
    sort(a.begin(),a.end());
    sort(b.begin(),b.end());
    cout<<(a==b?"YES":"NO");
}
```

6. String - Count Vowels

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    string s; cin>>s;
    int c=0;
    for(char x:s)
        if(string("aeiouAEIOU").find(x)!=string::npos) c++;
    cout<<c;
}
```

7. Linked List - Insert at Beginning

```
#include <bits/stdc++.h>
using namespace std;
struct Node{
    int data;
    Node* next;
};
int main(){
    Node* head=nullptr;
    Node* n=new Node{10,head};
    head=n;
    cout<<head->data;
}
```

8. Linked List - Reverse

```
#include <bits/stdc++.h>
using namespace std;
struct Node{
    int data;
    Node* next;
};
Node* reverse(Node* head){
    Node* prev=nullptr;
    while(head){
        Node* nxt=head->next;
        head->next=prev;
        prev=head;
        head=nxt;
    }
    return prev;
}
int main(){}
```

9. Linked List - Detect Loop

```
#include <bits/stdc++.h>
using namespace std;
struct Node{int data; Node* next;};
bool detect(Node* h){
    Node *s=h,*f=h;
    while(f&&f->next){
        s=s->next; f=f->next->next;
        if(s==f) return true;
    }
    return false;
}
int main(){}
```

10. Tree - Inorder Traversal

```
#include <bits/stdc++.h>
using namespace std;
struct Node{
    int data;
    Node* left;
    Node* right;
};
void inorder(Node* r){
    if(!r) return;
    inorder(r->left);
    cout<<r->data<<" ";
    inorder(r->right);
}
int main(){}
```

11. Tree - Height

```
int height(Node* r){
    if(!r) return 0;
    return 1+max(height(r->left),height(r->right));
}
```

12. Tree - Level Order

```
#include <queue>
void level(Node* r){
    queue<Node*> q;
    q.push(r);
    while(!q.empty()){
        Node* c=q.front(); q.pop();
        cout<<c->data<<" ";
        if(c->left) q.push(c->left);
        if(c->right) q.push(c->right);
    }
}
```

13. Graph - DFS

```
#include <bits/stdc++.h>
using namespace std;
vector<int> g[10];
bool vis[10];
void dfs(int u){
    vis[u]=1;
    cout<<u<<" ";
    for(int v:g[u])
        if(!vis[v]) dfs(v);
}
int main(){}
```

14. Graph - BFS

```
#include <bits/stdc++.h>
using namespace std;
vector<int> g[10];
void bfs(int s){
    queue<int> q;
    vector<int> vis(10,0);
    q.push(s); vis[s]=1;
    while(!q.empty()){
        int u=q.front(); q.pop();
        cout<<u<<" ";
        for(int v:g[u])
            if(!vis[v]){
                vis[v]=1;
                q.push(v);
            }
    }
}
```

15. Graph - Dijkstra

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n=5;
    vector<pair<int,int>> g[5];
    vector<int> dist(n,1e9);
    priority_queue<pair<int,int>, vector<pair<int,int>>, greater<>> pq;
    dist[0]=0;
    pq.push({0,0});
    while(!pq.empty()){
        auto [d,u]=pq.top(); pq.pop();
        for(auto [v,w]:g[u]){
            if(dist[v]>d+w){
                dist[v]=d+w;
                pq.push({dist[v],v});
            }
        }
    }
}
```

1. Array - Find Maximum

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n; cin>>n;
    int a[n];
    for(int i=0;i<n;i++) cin>>a[i];
    cout<<*max_element(a,a+n);
}
```

2. Array - Reverse

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n; cin>>n;
    int a[n];
    for(int i=0;i<n;i++) cin>>a[i];
    reverse(a,a+n);
    for(int i:a) cout<<i<<" ";
}
```

3. Array - Kadane

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n; cin>>n;
    int a[n];
    for(int i=0;i<n;i++) cin>>a[i];
    int cur=a[0], mx=a[0];
    for(int i=1;i<n;i++){
        cur=max(a[i],cur+a[i]);
        mx=max(mx,cur);
    }
    cout<<mx;
}
```

4. String - Palindrome

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    string s; cin>>s;
    string t=s;
    reverse(t.begin(),t.end());
    cout<<(s==t?"YES":"NO");
}
```

5. String - Anagram

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    string a,b; cin>>a>>b;
    sort(a.begin(),a.end());
    sort(b.begin(),b.end());
    cout<<(a==b?"YES":"NO");
}
```

6. String - Count Vowels

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    string s; cin>>s;
    int c=0;
    for(char x:s)
        if(string("aeiouAEIOU").find(x)!=string::npos) c++;
    cout<<c;
}
```

7. Linked List - Insert at Beginning

```
#include <bits/stdc++.h>
using namespace std;
struct Node{
    int data;
    Node* next;
};
int main(){
    Node* head=nullptr;
    Node* n=new Node{10,head};
    head=n;
    cout<<head->data;
}
```

8. Linked List - Reverse

```
#include <bits/stdc++.h>
using namespace std;
struct Node{
    int data;
    Node* next;
};
Node* reverse(Node* head){
    Node* prev=nullptr;
    while(head){
        Node* nxt=head->next;
        head->next=prev;
        prev=head;
        head=nxt;
    }
    return prev;
}
int main(){}
```

9. Linked List - Detect Loop

```
#include <bits/stdc++.h>
using namespace std;
struct Node{int data; Node* next;};
bool detect(Node* h){
    Node *s=h,*f=h;
    while(f&&f->next){
        s=s->next; f=f->next->next;
        if(s==f) return true;
    }
    return false;
}
int main(){}
```

10. Tree - Inorder Traversal

```
#include <bits/stdc++.h>
using namespace std;
struct Node{
    int data;
    Node* left;
    Node* right;
};
void inorder(Node* r){
    if(!r) return;
    inorder(r->left);
    cout<<r->data<<" ";
    inorder(r->right);
}
int main(){}
```

11. Tree - Height

```
int height(Node* r){
    if(!r) return 0;
    return 1+max(height(r->left),height(r->right));
}
```

12. Tree - Level Order

```
#include <queue>
void level(Node* r){
    queue<Node*> q;
    q.push(r);
    while(!q.empty()){
        Node* c=q.front(); q.pop();
        cout<<c->data<<" ";
        if(c->left) q.push(c->left);
        if(c->right) q.push(c->right);
    }
}
```

13. Graph - DFS

```
#include <bits/stdc++.h>
using namespace std;
vector<int> g[10];
bool vis[10];
void dfs(int u){
    vis[u]=1;
    cout<<u<<" ";
    for(int v:g[u])
        if(!vis[v]) dfs(v);
}
int main(){}
```

14. Graph - BFS

```
#include <bits/stdc++.h>
using namespace std;
vector<int> g[10];
void bfs(int s){
    queue<int> q;
    vector<int> vis(10,0);
    q.push(s); vis[s]=1;
    while(!q.empty()){
        int u=q.front(); q.pop();
        cout<<u<<" ";
        for(int v:g[u])
            if(!vis[v]){
                vis[v]=1;
                q.push(v);
            }
        }
    }
```

15. Graph - Dijkstra

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n=5;
    vector<pair<int,int>> g[5];
    vector<int> dist(n,1e9);
    priority_queue<pair<int,int>, vector<pair<int,int>>, greater<>> pq;
    dist[0]=0;
    pq.push({0,0});
    while(!pq.empty()){
        auto [d,u]=pq.top(); pq.pop();
        for(auto [v,w]:g[u]){
            if(dist[v]>d+w){
                dist[v]=d+w;
                pq.push({dist[v],v});
            }
        }
    }
}
```

1. Array - Find Maximum

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n; cin>>n;
    int a[n];
    for(int i=0;i<n;i++) cin>>a[i];
    cout<<*max_element(a,a+n);
}
```

2. Array - Reverse

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n; cin>>n;
    int a[n];
    for(int i=0;i<n;i++) cin>>a[i];
    reverse(a,a+n);
    for(int i:a) cout<<i<<" ";
}
```

3. Array - Kadane

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n; cin>>n;
    int a[n];
    for(int i=0;i<n;i++) cin>>a[i];
    int cur=a[0], mx=a[0];
    for(int i=1;i<n;i++){
        cur=max(a[i],cur+a[i]);
        mx=max(mx,cur);
    }
    cout<<mx;
}
```

4. String - Palindrome

```
#include <bits/stdc++.h>
using namespace std;
```

```
int main(){
    string s; cin>>s;
    string t=s;
    reverse(t.begin(),t.end());
    cout<<(s==t?"YES":"NO");
}
```

5. String - Anagram

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    string a,b; cin>>a>>b;
    sort(a.begin(),a.end());
    sort(b.begin(),b.end());
    cout<<(a==b?"YES":"NO");
}
```

6. String - Count Vowels

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    string s; cin>>s;
    int c=0;
    for(char x:s)
        if(string("aeiouAEIOU").find(x)!=string::npos) c++;
    cout<<c;
}
```

7. Linked List - Insert at Beginning

```
#include <bits/stdc++.h>
using namespace std;
struct Node{
    int data;
    Node* next;
};
int main(){
    Node* head=nullptr;
    Node* n=new Node{10,head};
    head=n;
    cout<<head->data;
}
```

8. Linked List - Reverse

```
#include <bits/stdc++.h>
using namespace std;
struct Node{
    int data;
    Node* next;
};
Node* reverse(Node* head){
    Node* prev=nullptr;
    while(head){
        Node* nxt=head->next;
        head->next=prev;
        prev=head;
        head=nxt;
    }
    return prev;
}
int main(){}
```

9. Linked List - Detect Loop

```
#include <bits/stdc++.h>
using namespace std;
```

```
struct Node{int data; Node* next;};
bool detect(Node* h){
    Node *s=h,*f=h;
    while(f&&f->next){
        s=s->next; f=f->next->next;
        if(s==f) return true;
    }
    return false;
}
int main(){}
```

10. Tree - Inorder Traversal

```
#include <bits/stdc++.h>
using namespace std;
struct Node{
    int data;
    Node* left;
    Node* right;
};
void inorder(Node* r){
    if(!r) return;
    inorder(r->left);
    cout<<r->data<<" ";
    inorder(r->right);
}
int main(){}
```

11. Tree - Height

```
int height(Node* r){
    if(!r) return 0;
    return 1+max(height(r->left),height(r->right));
}
```

12. Tree - Level Order

```
#include <queue>
void level(Node* r){
    queue<Node*> q;
    q.push(r);
    while(!q.empty()){
        Node* c=q.front(); q.pop();
        cout<<c->data<<" ";
        if(c->left) q.push(c->left);
        if(c->right) q.push(c->right);
    }
}
```

13. Graph - DFS

```
#include <bits/stdc++.h>
using namespace std;
vector<int> g[10];
bool vis[10];
void dfs(int u){
    vis[u]=1;
    cout<<u<<" ";
    for(int v:g[u])
        if(!vis[v]) dfs(v);
}
int main(){}
```

14. Graph - BFS

```
#include <bits/stdc++.h>
using namespace std;
vector<int> g[10];
void bfs(int s){
    queue<int> q;
    vector<int> vis(10,0);
    q.push(s); vis[s]=1;
    while(!q.empty()){
        int u=q.front(); q.pop();
        cout<<u<<" ";
        for(int v:g[u])
            if(!vis[v]){
                vis[v]=1;
                q.push(v);
            }
    }
}
```

15. Graph - Dijkstra

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n=5;
    vector<pair<int,int>> g[5];
    vector<int> dist(n,1e9);
    priority_queue<pair<int,int>, vector<pair<int,int>>, greater<>> pq;
    dist[0]=0;
    pq.push({0,0});
    while(!pq.empty()){
        auto [d,u]=pq.top(); pq.pop();
        for(auto [v,w]:g[u]){
            if(dist[v]>d+w){
                dist[v]=d+w;
                pq.push({dist[v],v});
            }
        }
    }
}
```

1. Array - Find Maximum

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n; cin>>n;
    int a[n];
    for(int i=0;i<n;i++) cin>>a[i];
    cout<<*max_element(a,a+n);
}
```

2. Array - Reverse

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n; cin>>n;
    int a[n];
    for(int i=0;i<n;i++) cin>>a[i];
    reverse(a,a+n);
    for(int i:a) cout<<i<<" ";
}
```

3. Array - Kadane

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n; cin>>n;
    int a[n];
    for(int i=0;i<n;i++) cin>>a[i];
    int cur=a[0], mx=a[0];
    for(int i=1;i<n;i++){
        cur=max(a[i],cur+a[i]);
        mx=max(mx,cur);
    }
    cout<<mx;
}
```

4. String - Palindrome

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    string s; cin>>s;
    string t=s;
    reverse(t.begin(),t.end());
    cout<<(s==t?"YES":"NO");
}
```

5. String - Anagram

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    string a,b; cin>>a>>b;
    sort(a.begin(),a.end());
    sort(b.begin(),b.end());
    cout<<(a==b?"YES":"NO");
}
```