1. A warehouse system stores package IDs in the order they arrive. To prepare for dispatch, the IDs must be sorted in ascending order. Write a program using Bubble Sort to arrange the following IDs: [5, 4, 3, 2, 1]

```
#include <iostream>
using namespace std;
int main() {
    int a[5] = {5,4,3,2,1}, n = 5;
    for(int i=0;i<n-1;i++)
        for(int j=0;j<n-i-1;j++)
            if(a[j]>a[j+1])
                swap(a[j],a[j+1]);
    cout<<"Sorted IDs: ";
    for(int i=0;i<n;i++) cout<<a[i]<<" ";
}
```

2. A warehouse system stores package IDs in the order they arrive. To prepare for dispatch, the IDs must be sorted in ascending order. Write a program using Insertion Sort to arrange the following IDs: [5,4,3,2,1]

```
#include <iostream>
using namespace std;
int main() {
    int a[5]={5,4,3,2,1}, n=5;
    for(int i=1;i<n;i++){
        int key=a[i], j=i-1;
        while(j>=0 && a[j]>key){
            a[j+1]=a[j];
            j--;
        }
        a[j+1]=key;
    }
    cout<<"Sorted IDs: ";
    for(int i=0;i<n;i++) cout<<a[i]<<" ";
}
```

3. A warehouse system stores package IDs in the order they arrive. To prepare for dispatch, the IDs must be sorted in ascending order. Write a program using Selection Sort to arrange the following IDs: [5,4,3,2,1]

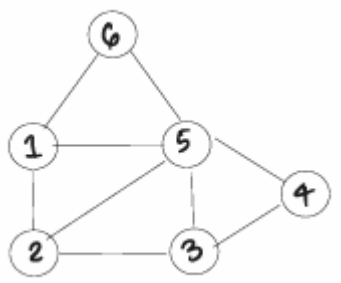
```
#include <iostream>
using namespace std;
int main() {
    int a[5]={5,4,3,2,1}, n=5;
    for(int i=0;i<n-1;i++){
        int min=i;
        for(int j=i+1;j<n;j++)
            if(a[j]<a[min]) min=j;
        swap(a[i],a[min]);
    }
    cout<<"Sorted IDs: ";
```

```
for(int i=0;i<n;i++) cout<<a[i]<<" ";
}
```

4. A hospital management system stores patient IDs in a linked list to maintain their admission order. You are given the following sequence of patient IDs: 111 → 123 → 124 → NULL Write a program to create and display this linked list.

```
#include <iostream>
using namespace std;
struct Node {
    int data;
    Node* next;
};
int main() {
    Node *n1=new Node{111,NULL};
    Node *n2=new Node{123,NULL};
    Node *n3=new Node{124,NULL};
    n1->next=n2; n2->next=n3;
    Node* temp=n1;
    while(temp){
        cout<<temp->data<<" -> ";
        temp=temp->next;
    }
    cout<<"NULL";
}
```

5. A social networking app wants to represent user connections as a graph, where each user is a node and friendships are edges between them. Given a graph showing user connections, create the adjacency list representation for it. Create the adjacency matrix for the given graph.




```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    int n=6;
    vector<int> adj[7];
    adj[1]={2,5,6};
    adj[2]={1,3,5};
    adj[3]={2,4,5};
```

```
    adj[4]={3,5};
    adj[5]={1,2,3,4,6};
    adj[6]={1,5};
    cout<<"Adjacency List:\n";
    for(int i=1;i<=n;i++){
        cout<<i<<" -> ";
        for(int j:adj[i]) cout<<j<<" ";
        cout<<"\n";
    }
    cout<<"\nAdjacency Matrix:\n";
    int mat[7][7]={0};
    for(int i=1;i<=n;i++)
        for(int j:adj[i]) mat[i][j]=1;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++) cout<<mat[i][j]<<" ";
        cout<<"\n";
    }
}
```

6. A university's examination system stores student roll numbers in a binary tree for efficient searching. Given the structure of the tree, implement and display the binary tree.

50 / \ 30 70 / \ / 20 40 60

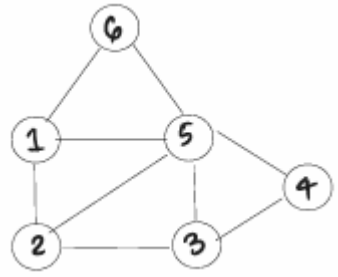
```
 #include <iostream>
using namespace std;
struct Node {
    int data;
    Node *left, *right;
};
Node* newNode(int d){
    Node* node=new Node();
    node->data=d;
    node->left=node->right=NULL;
    return node;
}
void inorder(Node* root){
    if(root){
        inorder(root->left);
        cout<<root->data<<" ";
        inorder(root->right);
    }
}
int main(){
    Node* root=newNode(50);
    root->left=newNode(30);
    root->right=newNode(70);
    root->left->left=newNode(20);
    root->left->right=newNode(40);
```

```
    root->right->left=newNode(60);
    cout<<"Inorder Traversal: ";
    inorder(root);
}
```

7. An online library wants to store book IDs efficiently using hashing. The hash function used is: h(key) = key % table_size If the book IDs are [1, 2, 3, 4] and the hash table size is 3, insert the keys into the hash table and show the final table representation

```
 #include <iostream>
 using namespace std;
 int main(){
    int a[]={1,2,3,4}, size=3, h[3]={-1,-1,-1};
    for(int x:a)
h[x%size]=x;
    for(int i=0;i<size;i++)
cout<<i<<":"<<h[i]<<" ";
 }
```

8. A city traffic control system represents road connections between intersections as a graph, where each intersection is a node and roads are edges. Given the graph, create the adjacency matrix representation for it.




```
#include <iostream>
using namespace std;
int main(){
    int n=6;
    int a[7][7]={0};
    int edges[][2]={{1,2},{1,5},{1,6},{2,3},{2,5},{3,4},{3,5},{4,5},{5,6}};
    int e=9;
    for(int i=0;i<e;i++){
        int u=edges[i][0], v=edges[i][1];
        a[u][v]=a[v][u]=1;
    }
    cout<<"Adjacency Matrix:\n";
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++) cout<<a[i][j]<<" ";
        cout<<"\n";
    }
}
```