QB_PBL - C++ Programs (Easy Version)

Q1. Bubble Sort

Question: 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]

Answer:

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

int main() {
    int arr[] = {5, 4, 3, 2, 1};
    int n = 5;

    for(int i = 0; i < n-1; i++) {
        for(int j = 0; j < n-i-1; j++) {
            if(arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }

    cout << "Sorted IDs: ";
    for(int i = 0; i < n; i++)
        cout << arr[i] << " ";
    return 0;
}
```

Q2. Insertion Sort

Question: Write a program using Insertion Sort to arrange the following IDs: [5, 4, 3, 2, 1]

Answer:

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

int main() {
    int arr[] = {5, 4, 3, 2, 1};
    int n = 5;

    for(int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;

        while(j >= 0 && arr[j] > key) {
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = key;
    }

    cout << "Sorted IDs: ";
    for(int i = 0; i < n; i++)
        cout << arr[i] << " ";
    return 0;
}
```

Q3. Selection Sort

Question: Write a program using Selection Sort to arrange the following IDs: [5, 4, 3, 2, 1]

Answer:

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

int main() {
    int arr[] = {5, 4, 3, 2, 1};
    int n = 5;

    for(int i = 0; i < n-1; i++) {
        int minIndex = i;
        for(int j = i+1; j < n; j++) {
            if(arr[j] < arr[minIndex])
                minIndex = j;
        }
        int temp = arr[i];
        arr[i] = arr[minIndex];
        arr[minIndex] = temp;
    }

    cout << "Sorted IDs: ";
    for(int i = 0; i < n; i++)
        cout << arr[i] << " ";
    return 0;
}
```

Q4. Linked List

Question: Create and display a linked list for patient IDs: 111 → 123 → 124 → NULL

Answer:

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

struct Node {
    int data;
    Node* next;
};

int main() {
    Node* head = new Node{111, nullptr};
    Node* second = new Node{123, nullptr};
    Node* third = new Node{124, nullptr};

    head->next = second;
    second->next = third;

    cout << "Linked List: ";
    Node* temp = head;
    while(temp != nullptr) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL";
    return 0;
}
```

Q5. Graph Representation

Question: Create the adjacency list and adjacency matrix representation for a given graph.

Answer:

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

int main() {
    int V = 4; // number of nodes
    vector<int> adj[4];

    adj[0] = {1, 2};
    adj[1] = {0, 2};
    adj[2] = {0, 1, 3};
    adj[3] = {2};

    cout << "Adjacency List:\n";
    for(int i = 0; i < V; i++) {
        cout << i << ": ";
        for(int j : adj[i])
            cout << j << " ";
        cout << endl;
    }

    cout << "\nAdjacency Matrix:\n";
    int matrix[4][4] = {0};
    matrix[0][1] = matrix[1][0] = 1;
    matrix[0][2] = matrix[2][0] = 1;
    matrix[1][2] = matrix[2][1] = 1;
    matrix[2][3] = matrix[3][2] = 1;

    for(int i = 0; i < V; i++) {
        for(int j = 0; j < V; j++)
            cout << matrix[i][j] << " ";
        cout << endl;
    }
    return 0;
}
```

Q6. Binary Tree

Question: Implement and display the binary tree: 50 $\rightarrow$ (30, 70), 30 $\rightarrow$ (20, 40), 70 $\rightarrow$ (60)

Answer:

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

struct Node {
    int data;
    Node *left, *right;
};

Node* newNode(int val) {
    Node* node = new Node;
    node->data = val;
    node->left = node->right = NULL;
    return node;
}

void inorder(Node* root) {
    if(root == NULL) return;
    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);
    return 0;
}
```

Q7. Hash Table

Question: Insert keys [1, 2, 3, 4] into hash table using h(key) = key % 3

Answer:

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

int main() {
    int keys[] = {1, 2, 3, 4};
    int size = 3;
    int hashTable[3] = {-1, -1, -1};

    for(int i = 0; i < 4; i++) {
        int index = keys[i] % size;
        hashTable[index] = keys[i];
    }

    cout << "Hash Table:\n";
    for(int i = 0; i < size; i++)
        cout << i << " -> " << hashTable[i] << endl;

    return 0;
}
```

Q8. Adjacency Matrix (City Traffic)

Question: Represent road connections between intersections using an adjacency matrix.

Answer:

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

int main() {
    int n = 4;
    int graph[4][4] = {
        {0, 1, 1, 0},
        {1, 0, 1, 0},
        {1, 1, 0, 1},
        {0, 0, 1, 0}
    };

    cout << "Adjacency Matrix:\n";
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < n; j++)
            cout << graph[i][j] << " ";
        cout << endl;
    }
    return 0;
}
```