ASSIGNMENT-LAB 04

Course Code: CSE - 2322 **Course Title:** Data Structures (& Lab)
Course Teacher: Mohammed Shamsul Alam

Name: Ezaz Ahmed **ID:** C223009 **Section:** 3AM

Problem 01: Write a program to sort n numbers using Insertion Sort algorithm.
Answer:
```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++)
    {
        cin >> arr[i];
    }
    for (int i = 1; i < n; i++)
    {
        int k = arr[i];
        int j = i - 1;

        while (j >= 0 && arr[j] > k)
        {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = k;
    }
    for (int i = 0; i < n; i++)
    {
        cout << arr[i] << " ";
    }
    return 0;
}
```
Problem 02: Write a program to sort n numbers using Selection Sort algorithm.
Answer:
```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++)
    {
        cin >> arr[i];
    }
    for (int i = 0; i < n - 1; i++)
    {
        int a = i;

        for (int j = i + 1; j < n; j++)
        {
            if (arr[j] < arr[a])
            {
                a = j;
            }
        }
        int temp = arr[a];
        arr[a] = arr[i];
        arr[i] = temp;
    }
    for (int i = 0; i < n; i++)
    {
        cout << arr[i] << " ";
    }
    return 0;
}
```

<u>Problem 03:</u> Write a program to sort n numbers using the Quick Sort algorithm.

<u>Answer:</u>

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

int part(int arr[], int l, int h)
{
    int a = arr[h];
    int i = l - 1;

    for (int j = l; j <= h - 1; j++)
    {
        if (arr[j] <= a)
        {
            swap(arr[j],arr[i++]);
        }
    }

    swap(arr[i + 1], arr[h]);
    return i + 1;
}
```

```
void QS(int arr[], int l, int h)
{
    if (l < h)
    {
        int pi = part(arr, l, h);
        QS(arr, l, pi - 1);
        QS(arr, pi + 1, h);
    }
}

int main()
{
    int n;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++)
    {
        cin >> arr[i];
    }

    QS(arr, 0, n - 1);
    for (int i = 0; i < n; i++)
    {
        cout << arr[i] << " ";
    }

    return 0;
}
```

Problem 04: Write a program to merge two sorted list.

Answer:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int a, b;

    cout << "Enter the size of the first list: ";
    cin >> a;
    vector<int> list1(a);
    cout << "Enter the elements of the first sorted list:\n";
    for (int i = 0; i < a; ++i)
    {
        cin >> list1[i];
    }
    cout << "Enter the size of the second list: ";
    cin >> b;
    vector<int> list2(b);
    cout << "Enter the elements of the second sorted list:\n";
    for (int i = 0; i < b; ++i)
    {
        cin >> list2[i];
    }
    vector<int> m(a + b);
    int i =0,j=0,k=0;
    while (i < a && j < b)
    {
        if (list1[i] < list2[j])
        {
            m[k++] = list1[i++];
        }
        else
        {
            m[k++] = list2[j++];
        }
    }

    while (i < a)
    {
        m[k++] = list1[i++];
    }
    while (j < b)
    {
        m[k++] = list2[j++];
    }


    cout << "Output: ";
    for (int i = 0; i < m.size(); ++i)
    {
        cout << m[i] << " ";
    }

    return 0;
}
```

Problem 05: Write a program to sort n numbers using Merge Sort algorithm.

Answer:

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

int main()
{
    int n;
```

```
cout << "Enter the number of elements: ";
cin >> n;

vector<int> arr(n);

cout << "Enter the elements:\n";
for (int i = 0; i < n; i++)
{
    cin >> arr[i];
}
for (int sz = 1; sz < n; sz *= 2)
{
    for (int left = 0; left < n - 1; left += 2 * sz)
    {
        int right = min(left + 2 * sz - 1, n - 1);
        int mid = min(left + sz - 1, n - 1);
        int i = left;
        int j = mid + 1;
        int k = 0;

        vector<int> temp(right - left + 1);

        while (i <= mid && j <= right)
        {
            if (arr[i] <= arr[j])
            {
                temp[k++] = arr[i++];
            }
            else
            {
                temp[k++] = arr[j++];
            }
        }

        while (i <= mid)
        {
            temp[k++] = arr[i++];
        }

        while (j <= right)
        {
            temp[k++] = arr[j++];
        }
        k = 0;
        for (int l = left; l <= right; l++)
        {
            arr[l] = temp[k++];
            }
        }
    }
    cout << "Output: ";
    for (int i = 0; i < n; ++i)
    {
        cout << arr[i] << " ";
    }

    return 0;
}
```

<u>Problem 06:</u> Write a program to create a Binary Search Tree of n elements and then display the elements (preorder, inorder and postorder) of the tree. <u>Answer:</u>

```
#include <bits/stdc++.h>
using namespace std;
struct Node
{
    int data;
    Node* left;
    Node* right;

    Node(int value) : data(value), left(nullptr), right(nullptr)
{}
};
Node* insert(Node* root, int value)
{
    if (root == nullptr)
    {
        return new Node(value);
    }

    if (value < root->data)
    {
        root->left = insert(root->left, value);
    }
    else
    {
        root->right = insert(root->right, value);
    }

    return root;
}

void preorder(Node* root)
{
    if (root == nullptr)
    {
        return;
    }

    cout << root->data << " ";
    preorder(root->left);
    preorder(root->right);
}
void inorder(Node* root)
{
    if (root == nullptr)
    {
        return;
    }

    inorder(root->left);
    cout << root->data << " ";
    inorder(root->right);
}
void postorder(Node* root)
{
    if (root == nullptr)
    {
        return;
    }

    postorder(root->left);
    postorder(root->right);
    cout << root->data << " ";
}

int main()
{
    int n;

    cout << "Enter the number of elements: ";
    cin >> n;

    Node* root = nullptr;

    cout << "Enter the elements:\n";
    for (int i = 0; i < n; ++i)
    {
        int value;
        cin >> value;
        root = insert(root, value);
    }
    cout << "Preorder: ";
    preorder(root);
    cout <<endl;

    cout << "Inorder: ";
    inorder(root);
    cout <<endl;

    cout << "Postorder: ";
    postorder(root);
    cout <<endl;

    return 0;
}
```

<u>Problem 07:</u> Write a program to create a Binary Search Tree of n elements and then search an element from the tree.

<u>Answer:</u>

```
#include <bits/stdc++.h>
using namespace std;
struct Node
{
    int data;
    Node* left;
    Node* right;

    Node(int value) : data(value), left(nullptr), right(nullptr)
{}
};

Node* insert(Node* root, int value)
{
    if (root == nullptr)
    {
        return new Node(value);
    }

    if (value < root->data)
    {
        root->left = insert(root->left, value);
    }
    else
    {
        root->right = insert(root->right, value);
    }

    return root;
```

```
}

bool search(Node* root, int value)
{
    if (root == nullptr)
    {
        return false;
    }

    if (root->data == value)
    {
        return true;
    }
    else if (value < root->data)
    {
        return search(root->left, value);
    }
    else
    {
        return search(root->right, value);
    }
}

int main()
{
    int n, a;

    cout << "Enter the number of elements: ";
    cin >> n;

    Node* root = nullptr;

    cout << "Enter the elements: ";
    for (int i = 0; i < n; ++i)
    {
        int b;
        cin >> b;
        root = insert(root, b);
    }

    cout << "Enter the value to search: ";
    cin >> a;
    if (search(root, a))
    {
        cout << a << " is found.";
    }
    else
    {
        cout << a << " is not found.";
    }
    return 0;
}
```
Problem 08: **Write a program to create a Binary Search Tree of n elements and then delete an element from the tree.**

Answer:

```
#include <iostream>
using namespace std;
struct Node
{
   int data;
   Node *left;
   Node *right;

   Node(int value) : data(value), left(nullptr), right(nullptr) {}
};

Node *insert(Node *root, int value)
{
   if (root == nullptr)
   {
      return new Node(value);
   }

   if (value < root->data)
   {
      root->left = insert(root->left, value);
   }
   else
   {
      root->right = insert(root->right, value);
   }

   return root;
}
```

```
Node *del(Node *root, int value)
{
   if (root == nullptr)
   {
      return root;
   }

   if (value < root->data)
   {
      root->left = del(root->left, value);
   }
   else if (value > root->data)
   {
      root->right = del(root->right, value);
   }
   else
   {
      if (root->left == nullptr)
      {
         Node *temp = root->right;
         delete root;
         return temp;
      }
      else if (root->right == nullptr)
      {
         Node *temp = root->left;
         delete root;
         return temp;
      }
      Node *temp = root->right;
      while (temp->left != nullptr)
      {
         temp = temp->left;
      }
      root->data = temp->data;
      root->right = del(root->right, temp->data);
   }
   return root;
}

void show(Node *root)
{
   if (root == nullptr)
   {
      return;
   }

   show(root->left);
   cout << root->data << " ";
   show(root->right);
}

int main()
{
   int n, d;

   cout << "Enter the number of elements: ";
   cin >> n;

   Node *root = nullptr;

   cout << "Enter the elements:\n";
   for (int i = 0; i < n; i++)
   {
      int value;
      cin >> value;
      root = insert(root, value);
   }

   cout << "Enter the value to delete: ";
   cin >> d;
   root = del(root, d);
   cout << "Output: ";
```

```
    show(root);
    return 0;
}
```
__Problem 09:__ **Write a program to create a Maxheap of n elements and then display the elements of the heap.**

__**Answer:**__

```
#include <bits/stdc++.h>
using namespace std;
void HEAP(vector<int>& heap, int n, int i)
{
    int l = i;
    int left = 2 * i + 1;
    int r = 2 * i + 2;

    if (left < n && heap[left] > heap[l])
    {
        l = left;
    }

    if (r < n && heap[r] > heap[l])
    {
        l = r;
    }

    if (l != i)
    {
        swap(heap[i], heap[l]);
        HEAP(heap, n, l);
    }
}

void BH(vector<int>& heap, int n)
{
    for (int i = n / 2 - 1; i >= 0; i--)
    {
        HEAP(heap, n, i);
    }
}

void show(const vector<int>& heap)
{
   for (int i = 0; i < heap.size(); i++)
   {
      cout << heap[i] << " ";
   }
}

int main()
{
   int n;

   cout << "Enter the number of elements: ";
   cin >> n;

   vector<int> MH(n);

   cout << "Enter the elements:\n";
   for (int i = 0; i < n; ++i)
   {
      cin >> MH[i];
   }

   BH(MH, n);

   cout << "Max Heap elements: ";
   show(MH);

   return 0;
}
```

<u>Problem 10:</u> Write a program to create a Maxheap of n elements and then delete an element from the heap.

<u>Answer:</u>

```
#include <bits/stdc++.h>
```

```
using namespace std;

void HEAP(vector<int>& heap, int n, int i)
{
   int l = i;
   int left = 2 * i + 1;
   int r = 2 * i + 2;

   if (left < n && heap[left] > heap[l])
   {
      l = left;
   }

   if (r < n && heap[r] > heap[l])
   {
      l = r;
   }

   if (l != i)
   {
      swap(heap[i], heap[l]);
      HEAP(heap, n, l);
   }
}

void BH(vector<int>& heap, int n)
{
   for (int i = n / 2 - 1; i >= 0; i--)
   {
      HEAP(heap, n, i);
   }
}

void del(vector<int>& heap, int& n, int value)
{
   int ind = -1;
   for (int i = 0; i < n; i++)
```

```
    {
        if (heap[i] == value)
        {
            ind = i;
            break;
        }
    }

    if (ind == -1)
    {
        cout << "Element not found in the Max Heap.";
        return;
    }
    heap[ind] = heap[n - 1];
    n--;
    BH(heap, n);
}

void show(const vector<int>& heap)
{
    for (int i = 0; i < heap.size(); i++)
    {
        cout << heap[i] << " ";
    }
}

int main()
{
    int n;

    cout << "Enter the number of elements: ";
    cin >> n;

    vector<int> MH(n);

    cout << "Enter the elements:\n";
    for (int i = 0; i < n; ++i)
    {
        cin >> MH[i];
    }

    BH(MH, n);

    cout << "Max Heap elements: ";
    show(MH);

    int d;
    cout << "Enter the value to delete: ";
    cin >> d;

    del(MH, n, d);

    cout << "Max Heap elements after deletion: ";
    show(MH);

    return 0;
}
```

Problem 11: Write a program to sort n numbers using Heap sort algorithm.

Answer:

```
#include <bits/stdc++.h>
using namespace std;
void HEAP(vector<int>& heap, int n, int i)
{
    int largest = i;
    int l = 2 * i + 1;
    int r = 2 * i + 2;

    if (l < n && heap[l] > heap[largest])
    {
        largest = l;
    }

    if (r < n && heap[r] > heap[largest])
    {
        largest = r;
    }

    if (largest != i)
    {
        swap(heap[i], heap[largest]);
        HEAP(heap, n, largest);
    }
}

void BH(vector<int>& heap, int n)
{
    for (int i = n / 2 - 1; i >= 0; i--)
    {
        HEAP(heap, n, i);
    }
}

void HS(vector<int>& heap, int n)
{
    BH(heap, n);

    for (int i = n - 1; i > 0; i--)
    {
        swap(heap[0], heap[i]);
        HEAP(heap, i, 0);
    }
}

void show(const vector<int>& arr)
{
    for (int i = 0; i < arr.size(); i++)
    {
        cout << arr[i] << " ";
    }
}
```

```
int main()
{
   int n;
   cout << "Enter the number of elements: ";
   cin >> n;
   vector<int> arr(n);
   cout << "Enter the elements:\n";
   for (int i = 0; i < n; i++)
   {
      cin >> arr[i];
   }
   HS(arr, n);
   cout << "Output:  ";
   show(arr);
   return 0;
}
```

Problem 12: Write a program to display the adjacency matrix of a graph.

Answer:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
   int ver, ed;
   cout << "Enter the number of vertices: ";
   cin >> ver;
   cout << "Enter the number of edges: ";
   cin >> ed;
   vector<vector<int>> ADM(ver, vector<int>(ver, 0));

   cout << "Enter the pairs: ";

   for (int i = 0; i < ed; i++)
   {
      int v1, v2;
      cin >> v1 >> v2;
```

```
        ADM[v1][v2] = 1;
        ADM[v2][v1] = 1;
    }

    cout << "Adjacency Matrix:"<<endl;
    for (int i = 0; i < ver; i++)
    {
        for (int j = 0; j < ver; j++)
        {
            cout << ADM[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Problem 13: **Write a program to display the path matrix of a graph from an adjacency matrix.**

Answer:

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

void show(const vector<vector<int>>& matrix)
{
    for (const auto& row : matrix)
    {
        for (int value : row)
        {
            cout << value << " ";
        }
        cout << endl;
    }
}

int main()
{
    int ver;
    cout << "Enter the number of vertices: ";
    cin >> ver;

    vector<vector<int>> ADM(ver, vector<int>(ver));

    cout << "Enter the adjacency matrix:\n";
    for (int i = 0; i < ver; i++)
    {
        for (int j = 0; j < ver; j++)
        {
            cin >> ADM[i][j];
        }
    }

    vector<vector<int>> pmat = ADM;

    for (int k = 0; k < ver; ++k)
    {
        for (int i = 0; i < ver; ++i)
        {
            for (int j = 0; j < ver; ++j)
            {
                if (pmat[i][k] && pmat[k][j])
                {
                    pmat[i][j] = 1;
                }
            }
        }
    }

    cout << "Path Matrix: ";
    show(pmat);

    return 0;
}
```

<u>Problem 14:</u> Write a program to display the path matrix of a graph using Warshall's Algorithm.

<u>Answer:</u>

```
#include <bits/stdc++.h>
using namespace std;
void show(const vector<vector<int>>& matrix)
{
   for (const auto& row : matrix)
   {
      for (int value : row)
      {
         cout << value << " ";
      }
      cout << endl;
   }
}

int main()
{
   int ver;

   cout << "Enter the number of vertices: ";
   cin >> ver;

   vector<vector<int>> ADM(ver, vector<int>(ver));

   cout << "Enter the adjacency matrix:\n";
   for (int i = 0; i < ver; i++)
   {
      for (int j = 0; j < ver; j++)
      {
         cin >> ADM[i][j];
      }
   }
   for (int k = 0; k < ver; k++)
   {
      for (int i = 0; i < ver; i++)
```

```
        {
            for (int j = 0; j < ver; j++)
            {
                ADM[i][j] = ADM[i][j] || (ADM[i][k] && ADM[k][j]);
            }
        }
    }
    cout << "Path Matrix:\n";
    show(ADM);
    return 0;
}
```

Problem 15: Write a program to display the adjacency list of a graph.

Answer:

```
#include <bits/stdc++.h>
using namespace std;
void show(const vector<vector<int>>& ADM)
{
    for (int i = 0; i < ADM.size(); i++)
    {
        cout << "Vertex " << i << " -> ";
        for (int ne : ADM[i])
        {
            cout << ne << " ";
        }
        cout << endl;
    }
}

int main()
{
    int ver, ed;

    cout << "Enter the number of vertices: ";
    cin >> ver;

    cout << "Enter the number of edges: ";
    cin >> ed;
    vector<vector<int>> ADM(ver);
    cout << "Enter the edges (vertex pairs):\n";
    for (int i = 0; i < ed; ++i)
    {
        int v1, v2;
        cin >> v1 >> v2;
        ADM[v1].push_back(v2);
        ADM[v2].push_back(v1);
    }
    cout << "Adjacency List: ";
    show(ADM);
    return 0;
}
```

Problem 16: Write a program to traverse a graph using Breadth First Search.

Answer:

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

void bfs(const vector<vector<int>>& ADM, int sv)
{
    int ver = ADM.size();

    vector<bool> visited(ver, false);
    queue<int> q;

    q.push(sv);
    visited[sv] = true;

    cout << "BFS Traversal starting from vertex " << sv << ":"<<endl;

    while (!q.empty())
    {
        int cv = q.front();
        q.pop();

        cout << cv << " ";
        for (int ne : ADM[cv])
        {
            if (!visited[ne])
            {
                q.push(ne);
                visited[ne] = true;
            }
        }
    }
    cout << endl;
}

int main()
{
    int ver, ed;

    cout << "Enter the number of vertices: ";
    cin >> ver;

    cout << "Enter the number of edges: ";
    cin >> ed;

    vector<vector<int>> ADM(ver);

    cout << "Enter the edges (vertex pairs):\n";

    for (int i = 0; i < ed; ++i)
    {
        int v1, v2;
        cin >> v1 >> v2;
        ADM[v1].push_back(v2);
        ADM[v2].push_back(v1);
    }

    int sv;
    cout << "Enter the starting vertex for BFS: ";
    cin >> sv;
    bfs(ADM, sv);
```

```
   return 0;
}
```

Problem 17: **Write a program to traverse a graph using Depth First Search.**

Answer:

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

void dfs(const vector<vector<int>>& ADM, int sv, vector<bool>& visited)
{
   visited[sv] = true;
   cout << sv << " ";

   for (int ne : ADM[sv])
   {
      if (!visited[ne])
      {
         dfs(ADM, ne, visited);
      }
   }
}

void dfsTraversal(const vector<vector<int>>& ADM, int sv)
{
   int ver = ADM.size();
   vector<bool> visited(ver, false);
   cout << "DFS Traversal starting from vertex " << sv << ":\n";
   dfs(ADM, sv, visited);
   cout << endl;
}

int main()
{
   int ver, ed;
   cout << "Enter the number of vertices: ";
   cin >> ver;
   cout << "Enter the number of edges: ";
   cin >> ed;
```

```
    vector<vector<int>> ADM(ver);
    cout << "Enter the edges (vertex pairs):\n";
    for (int i = 0; i < ed; i++)
    {
        int v1, v2;
        cin >> v1 >> v2;
        ADM[v1].push_back(v2);
        ADM[v2].push_back(v1);
    }
    int sv;
    cout << "Enter the starting vertex for DFS: ";
    cin >> sv;
    dfsTraversal(ADM, sv);
    return 0;
}
```

Problem 18: **Write a program to implement a hash table using Division method & use linear probing for collision resolution.**

Answer:

```
#include <bits/stdc++.h>
using namespace std;
const int TABLE_SIZE = 10;
vector<int> table(TABLE_SIZE, -1);
int hashFunction(int key)
{
    return key % TABLE_SIZE;
}
void insert(int key)
{
    int index = hashFunction(key);
    while (table[index] != -1)
    {
        index = (index + 1) % TABLE_SIZE;
    }
    table[index] = key;
}
bool search(int key)
{
    int index = hashFunction(key);
    while (table[index] != -1)
    {
        if (table[index] == key)
        {
            return true;
        }
        index = (index + 1) % TABLE_SIZE;
    }
    return false;
}
void show()
{
    cout << "Hash Table: ";
    for (int i = 0; i < TABLE_SIZE; i++)
    {
        cout << i << ": " << table[i] << "\n";
    }
}

int main()
{
    insert(5);
    insert(15);
    insert(25);
    insert(35);
    insert(45);

    show();

    int keyS;
    cout << "Enter a key to search: ";
    cin >> keyS;

    if (search(keyS))
    {
        cout << "Key " << keyS << " found in the hash table.";
    }
    else
    {
        cout << "Key " << keyS << " not found in the hash table.";
    }
    return 0;
}
```

Problem 19: Write a program to implement a hash table using Division method & use double hashing for collision resolution.

Answer:

```
#include <bits/stdc++.h>
using namespace std;
const int TABLE_SIZE = 10;
vector<int> table(TABLE_SIZE, -1);
int HF(int key)
{
    return key % TABLE_SIZE;
}
int DHF(int key)
{
    const int SH = 7;
    return SH - (key % SH);
}

void insert(int key)
{
    int index = HF(key);
    int size = DHF(key);

    while (table[index] != -1)
    {
        index = (index + size) % TABLE_SIZE;
    }

    table[index] = key;
}
```

```
bool search(int key)
{
   int index = HF(key);
   int size = DHF(key);

   while (table[index] != -1)
   {
      if (table[index] == key)
      {
         return true;
      }

      index = (index + size) % TABLE_SIZE;
   }

   return false;
}

void show()
{
   cout << "Hash Table:\n";
   for (int i = 0; i < TABLE_SIZE; ++i)
   {
      cout << i << ": " << table[i] <<endl;
   }
}

int main()
{
   insert(5);
   insert(15);
   insert(25);
   insert(35);
   insert(45);
   show();
   int keyS;
   cout << "Enter a key to search: ";
```

```
    cin >> keyS;

    if (search(keyS))
    {
        cout << "Key " << keyS << " found in the hash table.";
    }
    else
    {
        cout << "Key " << keyS << " not found in the hash table.";
    }
    return 0;
}
```
Problem 20: **Write a program to implement a hash table using chaining method for collision Resolution.**

Answer:
```
#include<bits/stdc++.h>
using namespace std;
const int TABLE_SIZE = 10;
vector<list<int>> table(TABLE_SIZE);
int HF(int key)
{
    return key % TABLE_SIZE;
}
void insert(int key)
{
    int index = HF(key);
    table[index].push_back(key);
}
bool search(int key)
{
    int index = HF(key);
    for (int value : table[index])
    {
        if (value == key)
        {
            return true;
        }
    }
    return false;
}

void show()
{
    cout << "Hash Table:\n";
    for (int i = 0; i < TABLE_SIZE; i++)
    {
        cout << i << ": ";
        for (int value : table[i])
        {
            cout << value << " -> ";
        }
        cout << "null\n";
    }
}

int main()
{
    insert(5);
    insert(15);
    insert(25);
    insert(35);
    insert(45);

    show();

    int keyS;
    cout << "Enter a key to search: ";
    cin >> keyS;

    if (search(keyS))
    {
        cout << "Key " << keyS << " found in the hash table.";
    }
    else
    {
        cout << "Key " << keyS << " not found in the hash table.";
    }

    return 0;
}
```