ADVANCE DATA STRUCTURES AND ALGORITHM REPORT

ON

LAB PROGRAM'S

Submitted in the partial fulfilment of the requirements for
the award of the degree of

Master of Technology

In

Computer Engineering

[Cyber Security]

By

Shourya Gupta (324103202)

Department of Computer Engineering

National Institute of Technology Kurukshetra

CONTENT

CHAPTER 1

LIST OF EXPERIMENTS

1. Write a program to perform all basic data structure operations (Insertion, deletion, searching, sorting) over array. All must be independent function and the user must have a choice to select what he wants. Must be done in cpp and user input will be dynamic.

2. Design a code to perform given below operations in LinkedList.

(a) Insertion and deletion at any position

(b) Searching a given element and finds its location.

(c) Count the number of nodes in the linked list.

(d) Perform reverse operation in linked list.

3. Write a program multiply two polynomials using LinkedList.

4. Implement the code for enqueue and dequeue operations.

5. Write a program for tree traversing

(a)Inorder

(b)Preorder

(c)Postorder

6. Write a program to insert an element in binary search tree.

7. Write a program to sort the given elements with the sorting techniques given below.

(a) Merge Sort

(b) Quick sort

(c) Heap sort

8. Implement BFS and DFS traversing algorithms.

9. Write the implementation of matrix chain multiplication.

10. Write the code for LCS.

11. Find Minimum Cost Spanning Tree of a given connected undirected graph using Prim's algorithm.

CHAPTER 2

PROGRAMMING LOGIC IN C++

Answer 1>

```
/**
 *
 * Write a program to perform all basic data structure operations
 * (Insertion, deletion, searching, sorting) over array.
 *
 * Author: Shourya Gupta
 * Copyright: ShouryaBrahmastra
 *
 * **/

#include <iostream>
#include <malloc.h>
using namespace std;

void insertElement(int *arr, int &n, int element, int position);
void deleteElement(int *arr, int &n, int position);
int searchElement(int *arr, int n, int element);
void sortArray(int *data, int N);
void displayArray(int *arr, int n);

int main(void)
{
   int n = 0; // Current size of the array
   int N, choice, element, position;
   cout << "\nEnter the size of the array: ";
   cin >> N;
   int *arr = (int*)malloc(N * sizeof(int));

   while (true)
   {
      cout << "Choose an operation:" << endl;
      cout << "1. Insert an element" << endl;
      cout << "2. Delete an element" << endl;
      cout << "3. Search for an element" << endl;
      cout << "4. Sort the array" << endl;
      cout << "5. Display the array" << endl;
```

```
    cout << "6. Exit" << endl;
    cin >> choice;

    switch (choice)
    {
        case 1:
            cout << "Enter the element to insert: ";
            cin >> element;
            cout << "Enter the position: ";
            cin >> position;
            insertElement(arr, n, element, position);
            break;
        case 2:
            cout << "Enter the position of the element to delete: ";
            cin >> position;
            deleteElement(arr, n, position);
            break;
        case 3:
            cout << "Enter the element to search: ";
            cin >> element;
            position = searchElement(arr, n, element);
            if (position != -1)
            {
                cout << "Element found at position: " << position << endl;
            }
            else
            {
                cout << "Element not found!" << endl;
            }
            break;
        case 4:
            sortArray(arr, n);
            cout << "Array sorted." << endl;
            break;
        case 5:
            displayArray(arr, n);
            break;
        case 6:
            free(arr);
            return 0;
        default:
            cout << "Invalid choice!" << endl;
    }
}
```

```
}

void insertElement(int *arr, int &n, int element, int position)
{
   if (position > n || position < 0)
   {
      cout << "Invalid position!" << endl;
      return;
   }

   for (int i = n; i > position; i--)
   {
      arr[i] = arr[i - 1];
   }

   arr[position] = element;
   n++;
}

void deleteElement(int *arr, int &n, int position)
{
   if (position >= n || position < 0)
   {
      cout << "Invalid position!" << endl;
      return;
   }

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

   n--;
}

int searchElement(int *arr, int n, int element)
{
   for (int i = 0; i < n; i++)
   {
      if (arr[i] == element)
      {
         return i;
      }
   }
```

```
    return -1; // Element not found
}

void sortArray(int *data, int N)
{
    for (int i = 0; i < N-1; i++)
    {
        for (int j = 0; j < N-i-1; j++)
        {
            if (data[j] > data[j+1])
            {
                // Swap the elements
                int temp = data[j];
                data[j] = data[j+1];
                data[j+1] = temp;
            }
        }
    }
}

void displayArray(int *arr, int n)
{
    for (int i = 0; i < n; i++)
    {
        cout << arr[i] << " ";
    }
    cout << endl;
}
```

Answer 2>

```
/**
 *
 * Design a code to perform given below operations in LinkedList.
 *
 * (a) Insertion and deletion at any position
 * (b) Searching a given element and finds its location.
 * (c) Count the number of nodes in the linked list.
 * (d) Perform reverse operation in linked list.
 *
 * Author: Shourya Gupta
 * Copyright: ShouryaBrahmastra
 *
 * **/
#include <iostream>
using namespace std;

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

class LL
{
   Node* head;

   public:

   LL() : head(NULL) {}

   void insertAtBeginning(int value)
   {
      Node* new_node = new Node();
      new_node->data = value;
      new_node->next = head;
      head = new_node;
   }


   void insertAtEnd(int value)
   {
      Node* new_node = new Node();
      new_node->data = value;
      new_node->next = NULL;
```

```
    if (!head)
    {
        head = new_node;
        return;
    }


    Node* ptr = head;
    while (ptr->next)
    {
        ptr = ptr->next;
    }

    ptr->next = new_node;
}


void insertAtPosition(int value, int pos)
{
    Node* new_node = new Node();
    new_node->data = value;
    new_node->next=NULL;

    if (pos < 1)
    {
        cout << "Location must be greater or equal to 1." << endl;
        return;
    }

    if (pos == 1)
    {
        insertAtBeginning(value);
        return;
    }

    Node* temp = head;
    for (int i = 1; i < pos - 1; ++i)
    {
        temp = temp->next;
    }

    new_node->next = temp->next;
    temp->next = new_node;
}

void deletefrombegin()
{
    Node* temp = head;
```

```
    if (!head)
    {
        cout << "List is empty." << endl;
        return;
    }

    head = head->next;
    free(temp);
}

void deletefromend()
{
    Node* temp = head;
    if (!head)
    {
        cout << "List is empty." << endl;
        return;
    }

    if (!head->next)
    {
        free(head);
        head = NULL;
        return;
    }

    while (temp->next->next) {
        temp = temp->next;
    }

    free(temp->next);
    temp->next = NULL;
}

void deleteFromPosition(int pos)
{
    Node* temp = head;
    if (pos < 1)
    {
        cout << "Position should be greater or equal to 1." << endl;
        return;
    }

    if (pos == 1)
    {
        deletefrombegin();
        return;
    }
```

```
    for (int i = 1; i < pos - 1 && temp; ++i) {
        temp = temp->next;
    }

    if (!temp || !temp->next) {
        cout << "Position out of range." << endl;
        return;
    }
    // Save the node to be deleted
    Node* nodeToDelete = temp->next;
    // Update the next pointer
    temp->next = temp->next->next;
     // Delete the node
    delete nodeToDelete;
}

int search(int key)
{
    Node* current = head;
    int index = 0;

    while (current)
    {
        if (current->data == key)
        {
            return index;
        }
        current = current->next;
        index++;
    }
    return -1;
}
int count()
{
    Node* current = head;
    int count = 0;
    while (current)
    {
        count++;
        current = current->next;
    }
    return count;
}

void reverse() {
    Node* prev = nullptr;
    Node* current = head;
```

```
        while (current)
        {
            Node* nextNode = current->next;
            current->next = prev;
            prev = current;
            current = nextNode;
        }
        head = prev;
    }

    void printList()
    {
        Node* current = head;
        while (current) {
            cout << current->data << " -> ";
            current = current->next;
        }
        cout << "NULL" << endl;
    }

};

int main() {
    LL list;
    int choice, value, position;

    do
    {
        cout << "\nMenu:\n";
        cout << "1. Insert at Beginning\n";
        cout << "2. Insert at End\n";
        cout << "3. Insert at Position\n";
        cout << "4. Delete from Beginning\n";
        cout << "5. Delete from End\n";
        cout << "6. Delete from Position\n";
        cout << "7. Search for an Element\n";
        cout << "8. Count Nodes\n";
        cout << "9. Reverse List\n";
        cout << "10. Print List\n";
        cout << "0. Exit\n";
        cout << "\nEnter your choice: ";
        cin >> choice;

        switch (choice)
        {
            case 1:
                cout << "Enter value to insert at beginning: ";
```

```
            cin >> value;
            list.insertAtBeginning(value);
            break;
        case 2:
            cout << "Enter value to insert at end: ";
            cin >> value;
            list.insertAtEnd(value);
            break;
        case 3:
            cout << "Enter value to insert and position: ";
            cin >> value >> position;
            list.insertAtPosition(value, position);
            break;
        case 4:
            list.deletefrombegin();
            break;
        case 5:
            list.deletefromend();
            break;
        case 6:
            cout << "Enter position to delete from: ";
            cin >> position;
            list.deleteFromPosition(position);
            break;
        case 7:
            cout << "Enter value to search: ";
            cin >> value;
            position = list.search(value);
            if (position != -1) {
                cout << "Element found at position: " << position << endl;
            } else {
                cout << "Element not found." << endl;
            }
            break;
        case 8:
            cout << "Number of nodes: " << list.count() << endl;
            break;
        case 9:
            list.reverse();
            cout << "List reversed." << endl;
            break;
        case 10:
            list.printList();
            break;
        case 0:
            cout << "Exiting the program." << endl;
            break;
        default:
```

```
            cout << "Invalid choice. Please try again." << endl;
        }
    } while (choice != 0);

    return 0;
}
```

Answer 3>

```
/**
 *
 * Write a program multiply two polynomials using LinkedList.
 *
 * Author: Shourya Gupta
 * Copyright: ShouryaBrahmastra
 *
 * **/
#include <iostream>
using namespace std;

struct Node {
   int coeff, power;
   Node* next;
};

class Solution {
public:
   Node* createPolynomial(int degree) {
      Node* poly = NULL;
      for (int i = degree; i >= 0; --i) {
         int coeff;
         cout << "Enter coefficient for x^" << i << ": ";
         cin >> coeff;
         poly = addNode(poly, coeff, i);
      }
      return poly;
   }

   Node* addNode(Node* start, int coeff, int power) {
      Node* newNode = new Node{coeff, power, NULL};
      if (!start)
         return newNode;

      Node* ptr = start;
      while (ptr->next) {
         ptr = ptr->next;
      }
      ptr->next = newNode;
      return start;
   }

   Node* multiply(Node* poly1, Node* poly2) {
      Node* poly3 = NULL;
      for (Node* ptr1 = poly1; ptr1; ptr1 = ptr1->next) {
         for (Node* ptr2 = poly2; ptr2; ptr2 = ptr2->next) {
```

```
            int coeff = ptr1->coeff * ptr2->coeff;
            int power = ptr1->power + ptr2->power;
            poly3 = addNode(poly3, coeff, power);
        }
    }
    removeDuplicates(poly3);
    return poly3;
}

void removeDuplicates(Node* start) {
    Node *ptr1 = start, *ptr2, *dup;
    while (ptr1 && ptr1->next) {
        ptr2 = ptr1;
        while (ptr2->next) {
            if (ptr1->power == ptr2->next->power) {
                ptr1->coeff += ptr2->next->coeff;
                dup = ptr2->next;
                ptr2->next = ptr2->next->next;
                delete dup;
            } else {
                ptr2 = ptr2->next;
            }
        }
        ptr1 = ptr1->next;
    }
}

void printList(Node* ptr) {
    while (ptr) {
        cout << (ptr->coeff > 0 && ptr != ptr->next ? "+" : "") << ptr->coeff << "x^" << ptr->power;
        ptr = ptr->next;
    }
    cout << endl;
}

};

// Driver code
int main() {
    Solution solution;
    char choice;
    do {
        int degree1, degree2;

        cout << endl;
        cout << "Enter the degree of the first polynomial: ";
        cin >> degree1;
        Node* poly1 = solution.createPolynomial(degree1);
```

```
        cout << endl;
        cout << "Enter the degree of the second polynomial: ";
        cin >> degree2;
        Node* poly2 = solution.createPolynomial(degree2);

        cout << endl;
        cout << "First Polynomial: ";
        solution.printList(poly1);

        cout << "Second Polynomial: ";
        solution.printList(poly2);

        cout << endl;
        Node* poly3 = solution.multiply(poly1, poly2);
        cout << "Resultant Polynomial: ";
        solution.printList(poly3);

        cout << endl;
        cout << "Do you want to multiply another pair of polynomials? (y/n): ";
        cin >> choice;

    } while (choice == 'y' || choice == 'Y');

    return 0;
}
```

Answer 4>

```
/**
 *
 * Implement the code for enqueue and dequeue operations.
 *
 * Author: Shourya Gupta
 * Copyright: ShouryaBrahmastra
 *
 * **/
#include <iostream>
using namespace std;

#define MAX 1000

class Queue
{
   int front, rear;
   int arr[MAX];

public:
   Queue()
   {
      front = -1;
      rear = -1;
   }

   bool isFull()
   {
      return (rear == MAX - 1);
   }

   bool isEmpty()
   {
      return (front == -1 || front > rear);
   }

   void enqueue(int x)
   {
      if (isFull())
      {
         cout << "Queue is full\n";
         return;
      }
      if (isEmpty())
      {
         front = 0;
      }
```

```
        arr[++rear] = x;
        cout << x << " enqueued to queue\n";
    }

    int dequeue()
    {
        if (isEmpty())
        {
            cout << "Queue is empty\n";
            return -1;
        }
        int x = arr[front++];
        if (isEmpty())
        {
            front = -1;
            rear = -1;
        }
        return x;
    }

    void display()
    {
        if (isEmpty())
        {
            cout << "Queue is empty\n";
            return;
        }
        for (int i = front; i <= rear; i++)
        {
            cout << arr[i] << " ";
        }
        cout << endl;
    }
};

int main()
{
    Queue q;
    int choice, value;
    do
    {
        cout << "\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice)
        {
        case 1:
            cout << "Enter value to enqueue: ";
```

```
            cin >> value;
            q.enqueue(value);
            break;
        case 2:
            value = q.dequeue();
            if (value != -1)
            {
                cout << "Dequeued element: " << value << endl;
            }
            break;
        case 3:
            q.display();
            break;
        case 4:
            cout << "Exiting...\n";
            break;
        default:
            cout << "Invalid choice, please try again.\n";
        }
    } while (choice != 4);
    return 0;
}
```

Answer 5>

```
/**
 *
 * Write a program for tree traversing
 *
 * (a) Inorder
 *
 * (b) Preorder
 *
 * (c) Postorder
 *
 * Author: Shourya Gupta
 * Copyright: ShouryaBrahmastra
 *
 * **/
#include<iostream>
using namespace std;

struct tree
{
   int data;
   struct tree *left;
   struct tree *right;

};

struct tree *createnode(int data)
{
   struct tree *t;
   t=(struct tree *)malloc(sizeof(struct tree));
   t->data=data;
   t->left=NULL;
   t->right=NULL;
   return t;
};

class TreeFunc
{
   public:
      void PreOrder(struct tree *r)
      {
         if(r!=NULL)
         {
            printf(" %d ",r->data);
            PreOrder(r->left);
            PreOrder(r->right);
         }
```

```
    }

    void PostOrder(struct tree *r)
    {
      if(r!=NULL)
      {
        PostOrder(r->left);
        PostOrder(r->right);
        printf(" %d ",r->data);
      }
    }

    void InOrder(struct tree *r)
    {
      if(r!=NULL)
      {
        InOrder(r->left);
        printf(" %d ",r->data);
        InOrder(r->right);
      }
    }

};

int main()
{
  /*
          5[p]
        /      \
      3[p2]      6[p5]
     /   \     /   \
   2[p3]  4[p1] 7[p6]  8[p7]
  /
 1[p4]
 */
  struct tree *p=createnode(5);
  struct tree *p1=createnode(4);
  struct tree *p2=createnode(3);
  struct tree *p3=createnode(2);
  struct tree *p4=createnode(1);
  struct tree *p5=createnode(6);
  struct tree *p6=createnode(7);
  struct tree *p7=createnode(8);

  p->left=p2;
  p->right=p5;
  p2->left=p3;
  p2->right=p1;
```

```
    p3->left=p4;
    p5->left=p6;
    p5->right=p7;

    cout<<endl;
    TreeFunc obj;

    cout << "\nPreOrder Traversal: "; obj.PreOrder(p);
    cout << "\nPostOrder Traversal: "; obj.PostOrder(p);
    cout << "\nInOrder Traversal: "; obj.InOrder(p);

    return 0;
}
```

Answer 6>

```
/**
 *
 * Write a program to insert an element in binary search tree.
 *
 * Author: Shourya Gupta
 * Copyright: ShouryaBrahmastra
 *
 * **/
#include <iostream>
using namespace std;

struct TreeNode {
   int data;
   TreeNode* left;
   TreeNode* right;
};

TreeNode* createNode(int data) {
   TreeNode* newNode = new TreeNode();
   if (!newNode) {
      cout << "Memory error\n";
      return NULL;
   }
   newNode->data = data;
   newNode->left = newNode->right = NULL;
   return newNode;
}

TreeNode* insertNode(TreeNode* root, int data) {

   if (root == NULL) {
      root = createNode(data);
      return root;
   }

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

   return root;
}

void preOrder(TreeNode* temp) {
   if (temp == NULL) {
```

```
        return;
    }
    cout << temp->data << " ";
    preOrder(temp->left);
    preOrder(temp->right);
}

int main() {
    TreeNode* root = NULL;
    int values[] = { 50, 30, 20, 40, 70, 60, 80 };
    int n = sizeof(values)/sizeof(values[0]);

    for (int i = 0; i < n; i++) {
        root = insertNode(root, values[i]);
    }

    cout << "Pre-order traversal: ";
    preOrder(root);
    cout << endl;

    int valueToInsert;
    cout << "Enter a value to insert into the BST: ";
    cin >> valueToInsert;

    root = insertNode(root, valueToInsert);

    cout << "Pre-order traversal after insertion: ";
    preOrder(root);
    cout << endl;

    return 0;
}
```

Answer 7a>

```
/**
 *
 * Write a program to sort the given elements with the sorting techniques given below
 *
 * (a) Merge Sort
 *
 * Author: Shourya Gupta
 * Copyright: ShouryaBrahmastra
 *
 * **/
#include <iostream>
#include <cstdlib>
using namespace std;

void mergesort(int *data, int low, int high);
void merge(int *data, int low, int mid, int high);
void printarray(int *data, int size);

int main() {
   int N;

   cout << "\nEnter the size of the array:\n";
   cin >> N;

   int *data = new int[N];
   cout << "\nEnter the elements of the array:\n";
   for (int i = 0; i < N; i++) {
      cin >> data[i];
   }

   cout << "\nGiven array is:\n";
   printarray(data, N);

   mergesort(data, 0, N - 1);

   cout << "\nSorted array is:\n";
   printarray(data, N);

   delete[] data;
   return 0;
}

void printarray(int *data, int size) {
   for (int i = 0; i < size; i++) {
      cout << data[i] << " ";
   }
```

```
    cout << endl;
}

void mergesort(int *data, int low, int high) {
    if (low < high) {
        int mid = low + (high - low) / 2;
        mergesort(data, low, mid);
        mergesort(data, mid + 1, high);

        merge(data, low, mid, high);
    }
}

void merge(int *data, int low, int mid, int high) {
    int size = high - low + 1;
    int *temp = new int[size];
    int i = low, j = mid + 1, k = 0;


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


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


    while (j <= high) {
        temp[k++] = data[j++];
    }


    for (int p = 0; p < size; p++) {
        data[low + p] = temp[p];
    }

    delete[] temp;
}
```

Answer 7b>

```
/**
 *
 * Write a program to sort the given elements with the sorting techniques given below
 *
 * (b) Quick sort
 *
 * Author: Shourya Gupta
 * Copyright: ShouryaBrahmastra
 *
 * **/
#include <iostream>
using namespace std;

void printarray(int *data, int N);
void QuickSort(int *data, int low, int high);
int partition(int *data, int low, int high);
void swap(int *a, int *b);

int main() {
   int N;

   cout << "\nEnter the size of the array:\n";
   cin >> N;

   int *data = new int[N];
   cout << "\nEnter the elements of the array:\n";
   for (int i = 0; i < N; i++) {
      cin >> data[i];
   }

   cout << "\nGiven array is:\n";
   printarray(data, N);

   QuickSort(data, 0, N - 1);

   cout << "\nSorted array is:\n";
   printarray(data, N);

   delete[] data;

   return 0;
}

void printarray(int *data, int N)
{
   for (int i = 0; i < N; i++)
```

```
    {
        cout << data[i] << " ";
    }
    cout << endl;
}

void QuickSort(int *data, int low, int high)
{
    if (low < high) {
        int j = partition(data, low, high);
        QuickSort(data, low, j - 1);
        QuickSort(data, j + 1, high);
    }
}

int partition(int *data, int low, int high)
{
    int i = low + 1, j = high;
    int pivot = data[low];

    do
    {
        while (i <= high && data[i] < pivot)
        {
            i++;
        }
        while (j >= low && data[j] > pivot)
        {
            j--;
        }
        if (i < j)
        {
            swap(&data[i], &data[j]);
        }
    } while (i < j);

    swap(&data[low], &data[j]);

    return j;
}

void swap(int *a, int *b)
{
    *a = *a ^ *b ^ (*b = *a);
}
```

Answer 7c>

```
/**
 *
 * Write a program to sort the given elements with the sorting techniques given below
 *
 * (c) Heap sort
 *
 * Author: Shourya Gupta
 * Copyright: ShouryaBrahmastra
 *
 * **/
#include <iostream>
using namespace std;

void heapify(int *arr, int n, int i);
void heapSort(int *arr, int n);
void printArray(int *arr, int n);

int main() {
   int n;

   cout << "Enter the size of the array:\n";
   cin >> n;

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

   cout << "\nGiven array is:\n";
   printArray(arr, n);

   heapSort(arr, n);

   cout << "\nSorted array is:\n";
   printArray(arr, n);

   delete[] arr;
   return 0;
}

// Function to build a max heap
void heapify(int *arr, int n, int i) {
   int largest = i;
   int left = 2 * i + 1;
   int right = 2 * i + 2;
```

```
    if (left < n && arr[left] > arr[largest]) {
        largest = left;
    }


    if (right < n && arr[right] > arr[largest]) {
        largest = right;
    }


    if (largest != i)
    {
        swap(arr[i], arr[largest]);


        heapify(arr, n, largest);
    }
}

void heapSort(int *arr, int n) {

    for (int i = n / 2 - 1; i >= 0; i--) {
        heapify(arr, n, i);
    }


    for (int i = n - 1; i > 0; i--) {

        swap(arr[0], arr[i]);


        heapify(arr, i, 0);
    }
}

void printArray(int *arr, int n) {
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}
```

Answer 8>

```
/**
 *
 * Implement BFS and DFS traversing algorithms.
 *
 * Author: Shourya Gupta
 * Copyright: ShouryaBrahmastra
 *
 * **/
#include <iostream>
#include <vector>
#include <queue>
#include <stack>
using namespace std;

class Graph {
private:
   int vertices;
   vector<vector<int>> adjList;

public:

   Graph(int v) : vertices(v) {
      adjList.resize(vertices);
   }

   void addEdge(int u, int v) {
      adjList[u].push_back(v);
      adjList[v].push_back(u);
   }

   void BFS(int start) {
      vector<bool> visited(vertices, false);
      queue<int> q;

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

      cout << "BFS Traversal: ";
      while (!q.empty()) {
         int node = q.front();
         q.pop();
         cout << node << " ";

         for (int neighbor : adjList[node]) {
            if (!visited[neighbor]) {
               visited[neighbor] = true;
```

```
                q.push(neighbor);
            }
        }
    }
    cout << endl;
}


void DFSRecursive(int node, vector<bool> &visited) {
    visited[node] = true;
    cout << node << " ";

    for (int neighbor : adjList[node]) {
        if (!visited[neighbor]) {
            DFSRecursive(neighbor, visited);
        }
    }
}


void DFSIterative(int start) {
    vector<bool> visited(vertices, false);
    stack<int> st;

    st.push(start);

    cout << "DFS Traversal (Iterative): ";
    while (!st.empty()) {
        int node = st.top();
        st.pop();

        if (!visited[node]) {
            visited[node] = true;
            cout << node << " ";
        }

        for (auto it = adjList[node].rbegin(); it != adjList[node].rend(); ++it) {
            if (!visited[*it]) {
                st.push(*it);
            }
        }
    }
    cout << endl;
}


void DFS(int start) {
    vector<bool> visited(vertices, false);
```

```
        cout << "DFS Traversal (Recursive): ";
        DFSRecursive(start, visited);
        cout << endl;
    }
};

int main() {
    int vertices, edges;
    cout << "Enter the number of vertices and edges:\n";
    cin >> vertices >> edges;

    Graph g(vertices);

    cout << "Enter the edges (u v):\n";
    for (int i = 0; i < edges; i++) {
        int u, v;
        cin >> u >> v;
        g.addEdge(u, v);
    }

    int start;
    cout << "Enter the starting vertex:\n";
    cin >> start;

    g.BFS(start);
    g.DFS(start);
    g.DFSIterative(start);

    return 0;
}
```

Answer 9>

```
/**
 *
 * Write the implementation of matrix chain multiplication.
 *
 * Author: Shourya Gupta
 * Copyright: ShouryaBrahmastra
 *
 * **/
#include <iostream>
#include <climits>
using namespace std;

int matrixChainMultiplication(int *dims, int n) {

   int dp[n][n];


   for (int i = 1; i < n; i++) {
      dp[i][i] = 0;
   }


   for (int L = 2; L < n; L++)
   {
      for (int i = 1; i < n - L + 1; i++)
      {
         int j = i + L - 1;
         dp[i][j] = INT_MAX;


         for (int k = i; k < j; k++) {
            int cost = dp[i][k] + dp[k + 1][j] + dims[i - 1] * dims[k] * dims[j];
            if (cost < dp[i][j]) {
               dp[i][j] = cost;
            }
         }
      }
   }


   return dp[1][n - 1];
}

int main() {
   int n;
   cout << "Enter the number of matrices: ";
```

```
    cin >> n;

    int dims[n + 1];
    cout << "Enter the dimensions of the matrices:\n";
    for (int i = 0; i <= n; i++) {
        cin >> dims[i];
    }

    int minCost = matrixChainMultiplication(dims, n + 1);
    cout << "Minimum number of multiplications is: " << minCost << endl;

    return 0;
}
```

Answer 10>

```
/**
 *
 * Write the code for LCS.
 *
 * Author: Shourya Gupta
 * Copyright: ShouryaBrahmastra
 *
 * **/
#include <iostream>
#include <vector>
#include <string>
using namespace std;

int LCS(string str1, string str2)
{
   int n = str1.length();
   int m = str2.length();


   vector<vector<int>> dp(n + 1, vector<int>(m + 1, 0));


   for (int i = 1; i <= n; i++)
   {
      for (int j = 1; j <= m; j++)
      {
         if (str1[i - 1] == str2[j - 1])
         {
            dp[i][j] = dp[i - 1][j - 1] + 1;
         }
         else
         {
            dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
         }
      }
   }

   return dp[n][m];
}

string findLCS(string str1, string str2)
{
   int n = str1.length();
   int m = str2.length();
```

```
    vector<vector<int>> dp(n + 1, vector<int>(m + 1, 0));


    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= m; j++)
        {
            if (str1[i - 1] == str2[j - 1])
            {
                dp[i][j] = dp[i - 1][j - 1] + 1;
            }
            else
            {
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
            }
        }
    }


    string lcs = "";
    int i = n, j = m;
    while (i > 0 && j > 0)
    {
        if (str1[i - 1] == str2[j - 1])
        {
            lcs = str1[i - 1] + lcs;
            i--;
            j--;
        }
        else if (dp[i - 1][j] > dp[i][j - 1])
        {
            i--;
        }
        else
        {
            j--;
        }
    }

    return lcs;
}

int main()
{
    string str1, str2;

    cout << "Enter the first string: ";
    cin >> str1;
```

```
    cout << "Enter the second string: ";
    cin >> str2;

    int length = LCS(str1, str2);
    string lcs = findLCS(str1, str2);

    cout << "Length of Longest Common Subsequence: " << length << endl;
    cout << "Longest Common Subsequence: " << lcs << endl;

    return 0;
}
```

Answer 11>

```
/**
 *
 * Find Minimum Cost Spanning Tree of a given connected undirected graph using Prim's
algorithm.
 *
 * Author: Shourya Gupta
 * Copyright: ShouryaBrahmastra
 *
 * **/
#include <iostream>
#include <vector>
#include <climits>
using namespace std;

int findMinVertex(vector<int> &key, vector<bool> &inMST, int vertices)
{
   int minKey = INT_MAX, minIndex = -1;
   for (int v = 0; v < vertices; v++)
   {
      if (!inMST[v] && key[v] < minKey)
      {
         minKey = key[v];
         minIndex = v;
      }
   }
   return minIndex;
}

void primMST(vector<vector<int>> &graph, int vertices)
{

   vector<int> key(vertices, INT_MAX);
   vector<bool> inMST(vertices, false);
   vector<int> parent(vertices, -1);
   key[0] = 0;

   for (int count = 0; count < vertices - 1; count++)
   {
      int u = findMinVertex(key, inMST, vertices);
      inMST[u] = true;
      for (int v = 0; v < vertices; v++)
      {
         if (graph[u][v] && !inMST[v] && graph[u][v] < key[v])
         {
            key[v] = graph[u][v];
            parent[v] = u;
```

```
            }
        }
    }


    cout << "Edge \tWeight\n";
    for (int i = 1; i < vertices; i++)
    {
        cout << parent[i] << " - " << i << "\t" << graph[i][parent[i]] << endl;
    }
}

int main()
{
    int vertices;

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

    vector<vector<int>> graph(vertices, vector<int>(vertices, 0));

    cout << "Enter the adjacency matrix (0 if no edge):\n";
    for (int i = 0; i < vertices; i++)
    {
        for (int j = 0; j < vertices; j++)
        {
            cin >> graph[i][j];
        }
    }

    primMST(graph, vertices);

    return 0;
}
```

CHAPTER 3

SAMPLE OUTPUT

Program 1 sample output:-

Enter the size of the array: 4

Choose an operation:

1. Insert an element

2. Delete an element

3. Search for an element

4. Sort the array

5. Display the array

6. Exit

1

Enter the element to insert: 1

Enter the position: 0

Choose an operation:

1. Insert an element

2. Delete an element

3. Search for an element

4. Sort the array

5. Display the array

6. Exit

1

Enter the element to insert: 22

Enter the position: 1

Choose an operation:

1. Insert an element

2. Delete an element

3. Search for an element

4. Sort the array

5. Display the array

6. Exit

1

Enter the element to insert: 66

Enter the position: 2

Choose an operation:

1. Insert an element

2. Delete an element

3. Search for an element

4. Sort the array

5. Display the array

6. Exit

1

Enter the element to insert: 44

Enter the position: 3

Choose an operation:

1. Insert an element

2. Delete an element

3. Search for an element

4. Sort the array

5. Display the array

6. Exit

1

Enter the element to insert: 5

Enter the position: 4

Choose an operation:

1. Insert an element

2. Delete an element

3. Search for an element

4. Sort the array

5. Display the array

6. Exit

5

1 22 66 44 5

Choose an operation:

1. Insert an element

2. Delete an element

3. Search for an element

4. Sort the array

5. Display the array

6. Exit

4

Array sorted.

Choose an operation:

1. Insert an element

2. Delete an element

3. Search for an element

4. Sort the array

5. Display the array

6. Exit

5

1 5 22 44 66

Choose an operation:

1. Insert an element

2. Delete an element

3. Search for an element

4. Sort the array

5. Display the array

6. Exit

2

Enter the position of the element to delete: 44

Invalid position!

Choose an operation:

1. Insert an element

2. Delete an element

3. Search for an element

4. Sort the array

5. Display the array

6. Exit

5

1 5 22 44 66

Choose an operation:

1. Insert an element

2. Delete an element

3. Search for an element

4. Sort the array

5. Display the array

6. Exit

2

Enter the position of the element to delete: 2

Choose an operation:

1. Insert an element

2. Delete an element

3. Search for an element

4. Sort the array

5. Display the array

6. Exit

5

1 5 44 66

Choose an operation:

1. Insert an element

2. Delete an element

3. Search for an element

4. Sort the array

5. Display the array

6. Exit

3

Enter the element to search: 5

Element found at position: 1

Choose an operation:

1. Insert an element

2. Delete an element

3. Search for an element

4. Sort the array

5. Display the array

6. Exit

6

Program 2 sample output:-

Menu:

1. Insert at Beginning

2. Insert at End

3. Insert at Position

4. Delete from Beginning

5. Delete from End

6. Delete from Position

7. Search for an Element

8. Count Nodes

9. Reverse List

10. Print List

0. Exit

Enter your choice: 1

Enter value to insert at beginning: 5

Menu:

1. Insert at Beginning

2. Insert at End

3. Insert at Position

4. Delete from Beginning

5. Delete from End

6. Delete from Position

7. Search for an Element

8. Count Nodes

9. Reverse List

10. Print List

0. Exit

Enter your choice: 1

Enter value to insert at beginning: 2

Menu:

1. Insert at Beginning

2. Insert at End

3. Insert at Position

4. Delete from Beginning

5. Delete from End

6. Delete from Position

7. Search for an Element

8. Count Nodes

9. Reverse List

10. Print List

0. Exit

Enter your choice: 2

Enter value to insert at end: 9

Menu:

1. Insert at Beginning

2. Insert at End

3. Insert at Position

4. Delete from Beginning

5. Delete from End

6. Delete from Position

7. Search for an Element

8. Count Nodes

9. Reverse List

10. Print List

0. Exit

Enter your choice: 10

2 -> 5 -> 9 -> NULL

Menu:

1. Insert at Beginning

2. Insert at End

3. Insert at Position

4. Delete from Beginning

5. Delete from End

6. Delete from Position

7. Search for an Element

8. Count Nodes

9. Reverse List

10. Print List

0. Exit

Enter your choice: 3

Enter value to insert and position: 6 3

Menu:

1. Insert at Beginning

2. Insert at End

3. Insert at Position

4. Delete from Beginning

5. Delete from End

6. Delete from Position

7. Search for an Element

8. Count Nodes

9. Reverse List

10. Print List

0. Exit

Enter your choice: 10

2 -> 5 -> 6 -> 9 -> NULL

Menu:

1. Insert at Beginning

2. Insert at End

3. Insert at Position

4. Delete from Beginning

5. Delete from End

6. Delete from Position

7. Search for an Element

8. Count Nodes

9. Reverse List

10. Print List

0. Exit

Enter your choice: 4

Menu:

1. Insert at Beginning

2. Insert at End

3. Insert at Position

4. Delete from Beginning

5. Delete from End

6. Delete from Position

7. Search for an Element

8. Count Nodes

9. Reverse List

10. Print List

0. Exit

Enter your choice: 5

Menu:

1. Insert at Beginning

2. Insert at End

3. Insert at Position

4. Delete from Beginning

5. Delete from End

6. Delete from Position

7. Search for an Element

8. Count Nodes

9. Reverse List

10. Print List

0. Exit

Enter your choice: 10

5 -> 6 -> NULL

Menu:

1. Insert at Beginning

2. Insert at End

3. Insert at Position

4. Delete from Beginning

5. Delete from End

6. Delete from Position

7. Search for an Element

8. Count Nodes

9. Reverse List

10. Print List

0. Exit

Enter your choice: 7

Enter value to search: 6

Element found at position: 1

Menu:

1. Insert at Beginning

2. Insert at End

3. Insert at Position

4. Delete from Beginning

5. Delete from End

6. Delete from Position

7. Search for an Element

8. Count Nodes

9. Reverse List

10. Print List

0. Exit

Enter your choice: 7

Enter value to search: 1

Element not found.

Menu:

1. Insert at Beginning

2. Insert at End

3. Insert at Position

4. Delete from Beginning

5. Delete from End

6. Delete from Position

7. Search for an Element

8. Count Nodes

9. Reverse List

10. Print List

0. Exit

Enter your choice: 8

Number of nodes: 2

Menu:

1. Insert at Beginning

2. Insert at End

3. Insert at Position

4. Delete from Beginning

5. Delete from End

6. Delete from Position

7. Search for an Element

8. Count Nodes

9. Reverse List

10. Print List

0. Exit

Enter your choice: 9

List reversed.

Menu:

1. Insert at Beginning

2. Insert at End

3. Insert at Position

4. Delete from Beginning

5. Delete from End

6. Delete from Position

7. Search for an Element

8. Count Nodes

9. Reverse List

10. Print List

0. Exit

Enter your choice: 10

6 -> 5 -> NULL

Menu:

1. Insert at Beginning

2. Insert at End

3. Insert at Position

4. Delete from Beginning

5. Delete from End

6. Delete from Position

7. Search for an Element

8. Count Nodes

9. Reverse List

10. Print List

0. Exit

Enter your choice: 0

Exiting the program.

Program 3 sample output:-

Enter the degree of the first polynomial: 3

Enter coefficient for x^3: 2

Enter coefficient for x^2: 5

Enter coefficient for x^1: 1

Enter coefficient for x^0: 9

Enter the degree of the second polynomial: 3

Enter coefficient for x^3: 4

Enter coefficient for x^2: 9

Enter coefficient for x^1: 6

Enter coefficient for x^0: 2

First Polynomial: +2x^3+5x^2+1x^1+9x^0

Second Polynomial: +4x^3+9x^2+6x^1+2x^0

Resultant Polynomial: +8x^6+38x^5+61x^4+79x^3+97x^2+56x^1+18x^0

Do you want to multiply another pair of polynomials? (y/n): y

Enter the degree of the first polynomial: 2

Enter coefficient for x^2: 4

Enter coefficient for x^1: 6

Enter coefficient for x^0: 1

Enter the degree of the second polynomial: 8

Enter coefficient for x^8: 4

Enter coefficient for x^7: 3

Enter coefficient for x^6: 9

Enter coefficient for x^5: 4

Enter coefficient for x^4: 6

Enter coefficient for x^3: 8

Enter coefficient for x^2: 2

Enter coefficient for x^1: 0

Enter coefficient for x^0: 1

First Polynomial: +4x^2+6x^1+1x^0

Second Polynomial: +4x^8+3x^7+9x^6+4x^5+6x^4+8x^3+2x^20x^1+1x^0

Resultant Polynomial:
+16x^10+36x^9+58x^8+73x^7+57x^6+72x^5+62x^4+20x^3+6x^2+6x^1+1x^0

Do you want to multiply another pair of polynomials? (y/n): n

Program 4 sample output:-

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 1

Enter value to enqueue: 5

5 enqueued to queue

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 1

Enter value to enqueue: 4

4 enqueued to queue

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 1

Enter value to enqueue: 9

9 enqueued to queue

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 1

Enter value to enqueue: 7

7 enqueued to queue

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 3

5 4 9 7

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 2

Dequeued element: 5

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 2

Dequeued element: 4

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 3

9 7

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 4

Exiting...

Program 5 sample output:-

PreOrder Traversal: 5 3 2 1 4 6 7 8

PostOrder Traversal: 1 2 4 3 7 8 6 5

InOrder Traversal: 1 2 3 4 5 7 6 8

Program 6 sample output:-

Pre-order traversal: 50 30 20 40 70 60 80

Enter a value to insert into the BST: 90

Pre-order traversal after insertion: 50 30 20 40 70 60 80 90

Program 7 (a) sample output:-

Enter the size of the array:

5

Enter the elements of the array:

2

9

1

7

3

Given array is:

2 9 1 7 3

Sorted array is:

1 2 3 7 9

Program 7 (b) sample output:-

Enter the size of the array:

6

Enter the elements of the array:

7

2

9

4

1

6

Given array is:

7 2 9 4 1 6

Sorted array is:

1 2 4 6 7 9

Program 7 (c) sample output:-

Enter the size of the array:

7

Enter the elements of the array:

9

4

6

1

0

2

5

Given array is:

9 4 6 1 0 2 5

Sorted array is:

0 1 2 4 5 6 9

Program 8 sample output:-

Enter the number of vertices and edges:

6 7

Enter the edges (u v):

0 1

0 2

1 3

1 4

2 4

3 5

4 5

Enter the starting vertex:

0

BFS Traversal: 0 1 2 3 4 5

DFS Traversal (Recursive): 0 1 3 5 4 2

DFS Traversal (Iterative): 0 1 3 5 4 2

Program 9 sample output:-

Enter the number of matrices: 4

Enter the dimensions of the matrices:

10 5 20 10 5

Minimum number of multiplications is: 1500

Program 10 sample output:-

Enter the first string: aabcde

Enter the second string: abbcddeee

Length of Longest Common Subsequence: 5

Longest Common Subsequence: abcde

Program 11 sample output:-

Enter the number of vertices: 5

Enter the adjacency matrix (0 if no edge):

0 2 0 6 0

2 0 3 8 5

0 3 0 0 7

6 8 0 0 9

0 5 7 9 0

Edge Weight

0 - 1 2

1 - 2 3

0 - 3 6

1 - 4 5