```
    int arr[n];
    cout << "Enter array elements";
    for (int i=0 ; i<n ; i++)
        cin >> arr[i];
    cout << "Enter number of positions to rotate";
    cin >> k;
    left_Rotate (arr, n, k);
    cout << "Rotated Array : ";
    for (int i=0 ; i<n ; i++)
        cout << arr[i] << " ";
    return 0;
}
```

Output

Clear

```
Enter size of array: 7
Enter array elements: 23
25
82
73
46
89
90
Enter number of positions to rotate: 3
Rotated Array: 73 46 89 90 23 25 82
```

Question 02.

Write a C program to ~~po~~ using stack data structure to convert an infix expression to postfix and evaluate the postfix expression:

```
#include <iostream>
#include <stack>
#include <string>
#include <cctype>
#include <sstype>
using namespace std;
int precedence(char op) {
  if (op == '+' || op == '-') return 1;
  if (op == '*' || op == '/') return 2;
  return 0;
}
string infixTopostfix (string infix) {
  stack <char> s;
  string postfix = "";
  for(char ch, infix) {
    if(isspace(ch)) continue;
    if (isdigit(ch)) {
      postfix += ch;
      postfix += ' ';
    }
    else if(ch == '('){
      s.push(ch);
  }
```

```
while (iss >> token) {
if (isdigit(token[0])) {
    s.push(stoi(token));
} else {
  int b = s.top(); s.pop();
  int a = s.top(); s.pop();
  if (token == "+") s.push(a+b);
  else if (token == "-") s.push(a-b);
  else if (token == "*") s.push(a*b);
  else if (token == "/") s.push(a/b);
}
}
return s.top();

int main(){
  string infix;
  cout << "Enter infix expression (eg (3+4)*5-6);
  getline(cin, infix);
  string postfix = infixtopostfix(infix);
  cout << "Postfix " = << postfix << endl;
  cout << " Evaluation Result : " << ~~evaluate~~
          evaluate_Postfix(postfix) << endl;
```

Output

Clear

```
Enter infix expression (e.g. (3 + 4) * 5 - 6): 3*7(4+8)/5
Postfix: 3 7 4 8 + * 5 /
Evaluation Result: 16
```

```
int main() {
    int n, val;
    Node *root = nullptr;
    cout << "Enter number of nodes to insert in BST";
    cin >> n;
    cout << "Enter values";
    for (int i=0; i<n; i++) {
        cin >> val;
        root = insert(root, val);
    }
    cout << "In-order"; inorder(root);
    cout << endl;
    cout << "Pre-order"; preorder(root);
    cout << endl;
    cout << "Post-order"; postorder(root);
    cout << endl;
    return 0;
```

Output

Clear

```
Enter number of nodes to insert in BST: 5
Enter the values: 1
3
7
4
9
In-order: 1 3 4 7 9
Pre-order: 1 3 7 4 9
Post-order: 4 9 7 3 1
```

Question 03

Write a C program to construct a Binary Search Tree (BST) with integer inputs. Implement and demonstrate the following operations

- Insertion
- In-order traversal
- Pre-order traversal
- Post-order traversal

```
#include <iostream>
using namespace std;
struct Node {
    int data;
    Node* left;
    Node* right;
    Node(int value) {
        data = value;
        left = right = nullptr;
    }
};
Node* insert(Node* root, int value) {
    if (!root) return new Node(value);
    if (value < root->data) root->left = insert(root->left);
    else root->right = insert(root->right, value);
    return root;
}
```

```
else if (ch == ')') {
    while (!s.empty() && s.top() != '(') {
        postfix += s.top(); postfix += ' ';
        s.pop();
    }
    s.pop();
}
s.pop();
}else {
    while (!s.empty() && precendence(ch) <=
            precendence(s.top())) {
        postfix += s.top(); postfix += ' ';
        s.pop();
    }
    s.push(ch);
}
}
while (!s.empty()) {
    postfix += s.top(); postfix += ' ';
    s.pop()
}}
return postfix;
}
int evaluatePostfix(string postfix) {
    stack<int> s;
    istringstream iss(postfix);
    string token;
```

Question 05

Write a C program to implement Quick Sort. Accept an array of integers, sort it using Quick sort and count the number of comparisons and swap made

```
#include <iostream>
using namespace std;
int comparisons=0, swaps=0;
int partition (int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low-1;
    for (int j= low ; j <high ; j++) {
        comparisons++
        if (arr[j] < pivot) {
            i++;
            swap(arr[i], arr[j]);
            swaps++;
        }
    }
    swap (arr(i+1), arr[high]);
    swaps++;
    return i+1;
}
int main () {
    int n;
    cout << "Enter number of elements ";
    cin >> n;
    int* arr = new int [n];
```

```
cout << "Enter elements ";
for (int i = 0; i < n; i++)
cin >> arr[i];
cout << "Sorted array";
for (int i = 0; i < n; i++) cout << arr[i] << "
cout << "Swaps " << swaps << endl;
delete [] arr;
return 0;
```

Output

Clear

```
Enter number of elements: 5
Enter elements: 9
6
3
8
5
Sorted Array: 3 5 6 8 9
Comparisons: 7
Swaps: 5
```

Question 04

Write a C program to represent a undirected graph using an adjacency matrix. Implement both DFS and BFS traversals for the graph.

```
#include <iostream>
#include <vector>
#include <queue>
using namespace std;
void DFS(int v, vector<vector<int>>& adj,
vector<bool>& visited) {
  visited[v] = true;
  cout << v << " ";
  for (int i = 0; i < adj.size(); i++) {
    if (adj[v][i] && !visited[i]) {
      DFS(i, adj, visited);
    }
  }
}
void BFS(int start, vector<vector<int>>& adj) {
  vector<bool> visited(adj.size(), false);
  queue<int> q;
  q.push(start);
  visited[start] = true;
  while (!q.empty()) {
    int v = q.front(); q.pop();
    cout << v << " ";
    for (int i=0; i<adj.size(); i++) {
      if (adj[v][i] && !visited[i]) {
```

```
                visited[i] = true;
                q.push(i);
            }
        }
    }
}
int main() {
    int v, e, u, v;
    cout << "DFS";
    vector<bool> visited(v, false);
    DFS(0, adj, visited);
    cout << "\n BFS";
    BFS(0, adj);
    return 0;
```

Output

Clear

```
Enter number of vertices: 5
Enter number of edges: 4
Enter edges (u v) pairs:
0 1
0 2
1 3
1 4
Adjacency Matrix:
0 1 1 0 0
1 0 0 1 1
1 0 0 0 0
0 1 0 0 0
0 1 0 0 0
DFS: 0 1 3 4 2
BFS: 0 1 2 3 4
```

21ECC212L : Data Structures and Algorithm
RA2311043010082 - Kirthika. KS.

Question 1 : Array Manipulation
Problem statement :
Write a C program to perform left rotation of an array by 'k' positions. The program should accept n integers and a number k from the user, and display the rotated array.

```
#include <iostream>
using namespace std;
void leftRotate (int arr[], int n, int k){
    k%=n;
    int *temp = new int[n];
    for (int i = 0; i<n ; i++) {
        temp[i] = arr[(i + k) %n];
    }
    for (int i =0 ; i<n ; i++ ) {
        arr[i] = temp[i];
    }
    delete temp;
}
int main () {
    int n, k;
    cout << "Enter size of array";
    cin >> n;
```