Name : Amit Sureshchandra Kesarwani

FY MCA Sem – 1

Subject : Data Structure with c/c++

Experiments : 1 Implementation of Sorting Techniques

1. Bubble Sort

Aim: Implement program for Bubble sort.
Objective: To understand working of bubble sort algorithm and sort array elements if they are not in the right order.

Program:

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

void swap(int* a, int i, int j) {
    int t = a[i];
    a[i] = a[j];
    a[j] = t;
}

int main() {
    // size of array
    int n;

    cout<< "Enter the size of array : ";
    cin >>n;
    int* a = new int[n];
    cout<< "Enter array elements " << endl;

    for(int i=0; i< n; i++) cin>> a[i];

    // bubble sort
    for(int i=0; i< n-1; i++) {
        for(int j=i+1; j< n; j++) if(a[i] > a[j]) swap(a, i, j);
    }

    cout<<"After Sorting"<< endl;
    for(int i=0; i< n; i++){
        cout<<a[i]<< " ";
    }
    return 0;
}
```

```
● → cpp_code git:(main) ✗ g++ p1/BubbleSort.cpp && ./a.out
Enter the size of array : 5
Enter array elements
3 4 1 2 7
After Sorting
1 2 3 4 7 %
○ → cpp_code git:(main) ✗
```

2. Insertion Sort

Aim: Implement program for Insertion sort.
Objective: To understand steps for sorting data using insertion sort algorithm. To implement program for sorting array elements using insertion sort.

Program:

```
● → cpp_code git:(main) ✗ g++ p1/InsertionSort.cpp && ./a.out
Enter the size of array : 8
Enter array elements
20 30 10 23 23 11 12 9
After Sorting
9 10 11 12 20 23 23 30
○ → cpp_code git:(main) ✗
```

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

void swap(int* a, int i, int j) {
    int t = a[i];
    a[i] = a[j];
    a[j] = t;
}
void print(int* a, int n) {
    for(int i=0; i< n; i++) cout<<a[i]<< " ";
}
int main() {
    // size of array
    int n;

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

    // insertion sort
    for(int i=1; i< n; i++) {
        for(int j=0; j< i; j++) if(a[i] < a[j]) swap(a, i, j);
    }
    cout<<"After Sorting"<< endl;
    print(a, n);
    return 0;
}
```

Experiments : 2 Implementation of Searching Techniques

Practical No: 1

Aim: Implement program for Linear Search.

Objective: Develop a program for searching an element from array using Linear search.

Program:

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

int linearSearch(int* a, int n, int target) {
    for(int i=0; i< n; i++)if(a[i] == target) return i;
    return -1;
}

int main() {
    // size of array
```

```
    int n;
    cout<< "Enter the size of array : ";
    cin >>n;
    int* a = new int[n];
    cout<< "Enter array elements " << endl;
    for(int i=0; i< n; i++)cin>> a[i];

    int t;
    cout<<"Enter element to search : ";
    cin >> t;

    int pos = linearSearch(a, n, t);
    if(pos == -1) cout<<"Not Found";
    else cout<<"Found at "<<(pos + 1);

    return 0;
}
```

```
➜  cpp_code git:(main) ✗ g++ p2/LinearSearch.cpp && ./a.out
Enter the size of array : 7
Enter array elements
2 3 12 15 36 1 8
Enter element to search : 1
Found at 6
➜  cpp_code git:(main) ✗ g++ p2/LinearSearch.cpp && ./a.out
Enter the size of array : 4
Enter array elements
2 3 1 6
Enter element to search : 0
Not Found
➜  cpp_code git:(main) ✗
```

Practical No: 2

Aim: Implement program for Binary Search.

Objective: To understand working of Binary search algorithm and to implement program for searching an element using binary search.

Program:

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

int binarySearch(int* a, int n, int target) {
    int l = 0, r = n-1;
    while(l <= r) {
        int mid = l + (r-l)/2;
        if(a[mid] == target) return mid;
        if(a[mid] > target)r = mid - 1;
        else l = mid + 1;
    }
    return -1;
}

int main() {
    // size of array
    int n;
    cout<< "Enter the size of array : ";
    cin >>n;
    int* a = new int[n];
    cout<< "Enter array elements " << endl;
    for(int i=0; i< n; i++) cin>> a[i];
```

```
    int t;
    cout<<"Enter element to search : ";
    cin >> t;
    // sort if not sorted
    sort(a, a + n);
    cout<<"sorted array"<< endl;
    for(int i=0; i< n; i++)cout<<a[i]<< " ";
    cout<<endl;
    int pos = binarySearch(a, n, t);
    if(pos == -1) cout<<"Not Found";
    else cout<<"Found at "<<(pos + 1);
    return 0;
}
```

```
➜  cpp_code git:(main) ✗ g++ p2/BinarySearch.cpp && ./a.out
Enter the size of array : 4
Enter array elements
1 2 3 4
Enter element to search : 1
sorted array
1 2 3 4
Found at 1
➜  cpp_code git:(main) ✗ g++ p2/BinarySearch.cpp && ./a.out
Enter the size of array : 5
Enter array elements
1 2 3 4 5
Enter element to search : 6
sorted array
1 2 3 4 5
Not Found
```

Experiments : 3 Modulo Division

Aim: Implement program for Modulo Division.

Objective: To understand modulo division method in hashing with the help of example. To implement program for finding key location of elements using Modulo division.

Program:

```
#include<iostream>
#define size 10
using namespace std;

int* a = new int[size];

int moduloHash(int t) { return t % size; }

int search(int target) { return a[moduloHash(target)]; }

void printHash() {
    for(int i=0; i< size; i++) cout<< "key : " << i << "\t" << "value : " << a[i]<< endl;
}

int main() {
    for(int i=0; i< size; i++) a[i] = -1;   // init hash with -1
    int n; // size of array
    cout<< "Enter the size of array : ";
    cin >>n;

    cout<< "Enter array elements " << endl;
    for(int i=0; i< n;){
        int temp;
        cin>> temp;
        int hash = moduloHash(temp); // calculate hash for temp .. if already found then
reject
        if(a[hash] != -1) cout<<"Collision : Unable to insert "<< temp<< endl;
        else {
            a[hash] = temp;
            i++; // increment only if insert is successful
        }
```

```
    }
    cout<< "~~~ Hash Table ~~~"<< endl;
    printHash();
    int t;
    cout<<"Enter element to search : ";
    cin >> t;
    int val = search(t);
    if(val == t) cout<<"Found";
    else cout<<"Not Found";

    return 0;
}
```

```
➜  cpp_code git:(main) ✗ g++ p3/ModuloDivision.cpp && ./a.out
Enter the size of array : 3
Enter array elements
11
21
Collision : Unable to insert 21
23
33
Collision : Unable to insert 33
12
~~~ Hash Table ~~~
key : 0 value : -1
key : 1 value : 11
key : 2 value : 12
key : 3 value : 23
key : 4 value : -1
key : 5 value : -1
key : 6 value : -1
key : 7 value : -1
key : 8 value : -1
key : 9 value : -1
Enter element to search : 23
Found%
➜  cpp_code git:(main) ✗ g++ p3/ModuloDivision.cpp && ./a.out
Enter the size of array : 4
Enter array elements
1
2
34
56
~~~ Hash Table ~~~
key : 0 value : -1
key : 1 value : 1
key : 2 value : 2
key : 3 value : -1
key : 4 value : 34
key : 5 value : -1
key : 6 value : 56
key : 7 value : -1
key : 8 value : -1
key : 9 value : -1
Enter element to search : 31
Not Found%
➜  cpp_code git:(main) ✗
```

Experiments : 4 Singly Linked List

Praticle :

Objective: The objective of implementing a Singly Linked List is to demonstrate how a basic linked list works and how we can perform various operations on it.

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

class LinkedList {
    public:

    int data;
    LinkedList *next;

    LinkedList(int v, LinkedList* next) {
        data = v;
        this->next = next;
    }
};

int length = 0;
LinkedList* head, * tail;

void print(){
    for(LinkedList *temp = head; temp != NULL;temp = temp->next) cout<< temp->data << " ";
    cout<<endl;
}

int search(int v) {
    LinkedList *temp = head;
    for(int pos= 0; temp != NULL; pos++, temp = temp->next) if(v == temp->data) return pos;
    return -1;
}

void insert(int val, int pos) {
    if(pos > length) return;
    if(pos == 0){ // insert at start
        head = new LinkedList(val, head);
        if(tail == NULL) tail = head;
    }
    else if(pos == length){   // insert at last
        tail->next = new LinkedList(val, NULL);
        tail = tail->next;
    }else {
        LinkedList *temp = head;
        for(;pos > 0; pos--, temp = temp->next) temp = temp->next;
        temp->next = new LinkedList(val, temp->next);
    }
    length++;
}

void removeAt(int pos) {
    if(length == 0) {
        cout<<"empty so can't remove" << endl;
        return;
    }
```

```
    if(pos < 0) return;
    if(pos == 0) head = head->next;
    else {
        LinkedList *temp = head;
        for(;pos > 1; pos--, temp = temp->next);
        temp->next = temp->next == NULL ? NULL : temp->next->next;
    }
    length--;
}
void removeVal(int v) {
    int pos = search(v);
    if(pos == -1) cout<<"Not Found"<< endl;
    else removeAt(pos);
}

int main() {

    cout<<"~~~~~~~ Menu ~~~~~~~"<< endl;
    cout<<"1. insert at any pos"<< endl;
    cout<<"2. insert at start"<< endl;
    cout<<"3. insert at end"<< endl;
    cout<<"4. remove value"<< endl;
    cout<<"5. remove start"<< endl;
    cout<<"6. remove end"<< endl;
    cout<<"7. view"<< endl;
    cout<<"8. clear screen"<< endl;
    cout<<"9. exit"<< endl;

    while(1) {

        cout<<"Enter your choice: ";
        char choice;
        cin >> choice;

        switch (choice)
        {
        case '1':
            int temp;
            cout<<"Enter the value to insert : ";
            cin >> temp;

            int pos;
            cout<<"Enter the position (index 1 based) : ";
            cin >> pos;
            if(pos > length+1) cout<< "out of bound .. cannot insert at " << pos << endl;
            else insert(temp, --pos);
            break;

        case '2':
            cout<<"Enter the value to insert : ";
            cin >> temp;
            insert(temp, 0);
            break;

        case '3':
            cout<<"Enter the value to insert : ";
            cin >> temp;
```

```
            insert(temp, length == 0 ? 0 : length);
            break;

        case '4':
            cout<<"Enter the value to remove : ";
            cin >> temp;
            removeVal(temp);
            break;
        case '5':
            removeAt(0);
            break;
        case '6':
            removeAt(length-1);
            break;
        case '7':
            print();
            break;
        case '8':
            system("clear"); // clear the screen

            cout<<"\n~~~~~~~ Menu ~~~~~~~"<< endl;
            cout<<"1. insert at any pos"<< endl;
            cout<<"2. insert at start"<< endl;
            cout<<"3. insert at end"<< endl;
            cout<<"4. remove value"<< endl;
            cout<<"5. remove start"<< endl;
            cout<<"6. remove end"<< endl;
            cout<<"7. view"<< endl;
            cout<<"8. clear screen"<< endl;
            cout<<"9. exit"<< endl;
            break;
        case '9':
            exit(0);
        default:
            cout<<"Invalid Choice"<< endl;
            break;
        }
    }
    return 0;
}
```

```
● → cpp_code git:(main) ✗ g++ p4/SingleLinkedList.cpp && ./a.out
  ~~~~~~~ Menu ~~~~~~~
  1. insert at any pos
  2. insert at start
  3. insert at end
  4. remove value
  5. remove start
  6. remove end
  7. view
  8. clear screen
  9. exit
  Enter your choice: 2
  Enter the value to insert : 10
  Enter your choice: 1
  Enter the value to insert : 20
  Enter the position (index 1 based) : 2
  Enter your choice: 3
  Enter the value to insert : 30
  Enter your choice: 7
  10 20 30
  Enter your choice: 2
  Enter the value to insert : 0
  Enter your choice: 7
  0 10 20 30
  Enter your choice: 4
  Enter the value to remove : 20
  Enter your choice: 7
  0 10 30
  Enter your choice: 5
  Enter your choice: 7
  10 30
  Enter your choice: 6
  Enter your choice: 7
  10
  Enter your choice: 9
○ → cpp_code git:(main) ✗
```

Experiments : 5 Stack

Aim: Implement program for Stack using Arrays.

Objective: To understand stack operations and to develop a program for implementing stack using array.

Program :

```cpp
#include<iostream>
#define size 10

using namespace std;

int* stack = new int[size];
int top = -1;

void push(int temp) {
    if(top + 1 >= size)
        cout<<"Overflow" << endl;
    else
        stack[++top] = temp;
}

void pop(){
    if(top == -1) cout<<"Underflow"<< endl;
```

```
    else cout<<"popped : "<< stack[top--] << endl;
}

void print(){
    for(int i=0; i<= top; i++) cout<< stack[i] << " ";
    cout<< endl;
}

void printTop() {
    if(top == -1) return;
    cout<<"top : " << stack[top]<< endl;
}

void printMenu() {
    cout<<"~~~~~~ Menu ~~~~~~~" << endl;
    cout<<"1. push"<< endl;
    cout<<"2. pop"<< endl;
    cout<<"3. print"<< endl;
    cout<<"4. top"<< endl;

    cout<<"8. clear screen"<< endl;
    cout<<"9. exit"<< endl;
}

int main() {
    int temp;
    printMenu();

    while(1){
        char choice;
        cout<<"Choose your choice : ";
        cin >> choice;

        switch(choice) {
            case '1':
                cout<<"Enter data : ";
                cin>> temp;
                push(temp);
                break;
            case '2':
                pop();
                break;
            case '3':
                print();
                break;
            case '4':
                printTop();
                break;
            case '8':
                system("clear"); // clear the screen
                printMenu();
                break;
            case '9':
                exit(0);
            default:
                cout<<"Invalid Choice"<< endl;
                break;
```

```
➜  cpp_code git:(main) ✗ g++ p5/Stack.cpp && ./a.out
~~~~~~ Menu ~~~~~~~
1. push
2. pop
3. print
4. top
8. clear screen
9. exit
Choose your choice : 1
Enter data : 10
Choose your choice : 1
Enter data : 20
Choose your choice : 1
Enter data : 30
Choose your choice : 4
top : 30
Choose your choice : 2
popped : 30
Choose your choice : 3
10 20
Choose your choice : 2
popped : 20
Choose your choice : 2
popped : 10
Choose your choice : 2
Underflow
Choose your choice : 9
➜  cpp_code git:(main) ✗
```

```
        }
    }


    return 0;
}
```

Practical No: 2

Experiments : 6 Stack Applicaton

Aim: Implement program for Evaluation of Postfix Expression.

Objective: To develop program for Evaluation of Postfix Expression.

Program :

```
#include <iostream>
#include <stack>
#include <string>
#include <cmath>

using namespace std;

bool isOperator(char c){
    return c == '+' || c == '-' || c == '*' || c == '/' || c == '^';
}

int performOperation(char op, int operand1, int operand2)
{
    switch (op){
    case '+':
        return operand1 + operand2;
    case '-':
        return operand1 - operand2;
    case '*':
        return operand1 * operand2;
    case '/':
        if (operand2 == 0)
            throw runtime_error("Division by zero error!");
        return operand1 / operand2;
    case '^':
        return pow(operand1, operand2);
    default:
        throw runtime_error("Invalid operator!");
    }
}

int evaluatePostfixExpression(string expression)
{
    stack<int> s;
    int operand1, operand2, result;
```

```
    for (char c : expression)
    {
        if (isdigit(c)){
            s.push(c - '0');
        }
        else{
            operand2 = s.top();
            s.pop();
            operand1 = s.top();
            s.pop();
            result = performOperation(c, operand1, operand2);
            s.push(result);
        }
    }

    if (s.size() != 1){
        throw runtime_error("Invalid expression!");
    }

    return s.top();
}

int main()
{
    string postfix_expression;
    cout<<"Enter the expression : ";
    cin >> postfix_expression;

    try{
        int result = evaluatePostfixExpression(postfix_expression);
        cout << "The result of the expression is: " << result << endl;
    }
    catch (const exception &e){
        cerr << "Error: " << e.what() << endl;
    }
    return 0;
}
```

```
ak@DESKTOP-R11F3IS:~/Data-Structure-in-cpp$ g++ p6/PostFixEvaluation.cpp && ./a.out
Enter the expression : 12+5*
The result of the expression is: 15
```

Experiments : 7 Queue

Aim : Implementation of Linear Queue operations

Objective:
The circular queue solves the main limitation of the normal queue.
In a normal queue, after some insertion and deletion, there will be blank, unusable space.

Program :

```cpp
#include<iostream>
#define LEN 10
using namespace std;

int* q = new int[LEN];
int f = -1, b = -1; // f - front, b - back

void print(){
    if(f == -1) return;
    for(int i=b; i<= f; i++) cout<< q[i] << " ";
    cout<<endl;
}

void enqueue(int v) {
    if(f + 1 == LEN){
         cout<< "Overflow" << endl;
         return;
    }
    q[++f] = v;
    cout<< v << " enqueued" << endl;
    if(b == -1) b++;
}
int dequeue() {
    if(b == -1 || b > f){
        cout<< "Underflow" << endl;
        return -1;
    }
    int v = q[b++];
    cout<< v << " dequeued" << endl;
    // below line makes it optimised with circular
    // if(b > f) b = f = -1;
    return v;
}

void front() {
    if(f == -1) return;
    cout<<q[f]<< endl;
}

void rear() {
    if(b == -1) return;
    cout<<q[b]<< endl;
}

void printMenu() {
    cout<<"~~~~~~~ Menu ~~~~~~~"<< endl;
    cout<<"1. enqueue"<< endl;
    cout<<"2. dequeue"<< endl;
    cout<<"3. front"<< endl;
    cout<<"4. rear"<< endl;
    cout<<"7. view"<< endl;
    cout<<"8. clear screen"<< endl;
    cout<<"9. exit"<< endl;
}
int main() {
```

```
    printMenu();
    while(1) {
        cout<<"Enter your choice: ";
        char choice;
        cin >> choice;
        switch (choice){
            case '1':
                int temp;
                cout<<"Enter the value to enqueue :
";
                cin >> temp;
                enqueue(temp);
                break;
            case '2':
                dequeue();
                break;
            case '3':
                front();
                break;
            case '4':
                rear();
                break;
            case '7':
                print();
                break;
            case '8':
                system("clear"); // clear the screen
                printMenu();
                break;
            case '9':
                exit(0);
            default:
                cout<<"Invalid Choice"<< endl;
                break;
        }
    }
    return 0;
}
```

```
cpp_code git:(main) ✗ g++ p7/Queue.cpp && ./a.out
~~~~~~~ Menu ~~~~~~~
1. enqueue
2. dequeue
3. front
4. rear
7. view
8. clear screen
9. exit
Enter your choice: 1
Enter the value to enqueue : 10
10 enqueued
Enter your choice: 1
Enter the value to enqueue : 20
20 enqueued
Enter your choice: 1
Enter the value to enqueue : 30
30 enqueued
Enter your choice: 7
10 20 30
Enter your choice: 3
30
Enter your choice: 4
10
Enter your choice:
7
10 20 30
Enter your choice: 2
10 dequeued
Enter your choice: 2
20 dequeued
Enter your choice: 7
30
Enter your choice: 2
30 dequeued
Enter your choice: 2
Underflow
Enter your choice: 7

Enter your choice: 9
cpp_code git:(main) ✗
```

Experiments : 8 BST

Implementation of BST and its traversal techniques(Any one)
a. Inorder
b. Preorder
c. Postorder

Program:

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

class Node {
    public:
        int data;
```

```
        Node* left;
        Node* right;

        Node(int data, Node *l, Node *r) {
            this->data = data;
            left = l;
            right = r;
        }

        Node(int data) {
            this->data = data;
            left = NULL;
            right = NULL;
        }
};
Node* root;
Node* addNode(Node* p, Node* n) {
    if(p == NULL) return n;
    if(p->data > n->data) p->left = addNode(p->left, n);
    else p->right = addNode(p->right, n);
    return p;
}

void add(int v) {
    root = addNode(root, new Node(v));
}

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

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

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

void printMenu() {
    cout<<"~~~~~~~ Menu ~~~~~~~"<< endl;
    cout<<"1. Add node"<< endl;
    cout<<"2. preorder"<< endl;
    cout<<"3. inorder"<< endl;
    cout<<"4. postorder"<< endl;
    cout<<"8. clear screen"<< endl;
    cout<<"9. exit"<< endl;
}
```

```cpp
int main() {
    printMenu();
    while(1) {
        cout<<"Enter your choice: ";
        char choice;
        cin >> choice;
        switch (choice){
            case '1':
                int temp;
                cout<<"Enter the node value to add : ";
                cin >> temp;
                add(temp);
                break;
            case '2':
                cout<< "Preorder : ";
                preorder(root);
                cout<< endl;
                break;
            case '3':
                cout<< "Inorder : ";
                inorder(root);
                cout<< endl;
                break;
            case '4':
                cout<< "Postorder : ";
                postorder(root);
                cout<< endl;
                break;
            case '8':
                system("clear"); // clear the screen
                printMenu();
                break;
            case '9':
                exit(0);
            default:
                cout<<"Invalid Choice"<< endl;
                break;
        }
    }
    return 0;
}
```

```
● → cpp_code git:(main) ✗ g++ p8/BST.cpp && ./a.out
~~~~~~~~ Menu ~~~~~~~~
1. Add node
2. preorder
3. inorder
4. postorder
8. clear screen
9. exit
Enter your choice: 1
Enter the node value to add : 10
Enter your choice: 1
Enter the node value to add : 5
Enter your choice: 1
Enter the node value to add : 20
Enter your choice: 1
Enter the node value to add : 15
Enter your choice: 2
Preorder : 10 5 20 15
Enter your choice: 3
Inorder : 5 10 15 20
Enter your choice: 4
Postorder : 5 15 20 10
Enter your choice: 9
○ → cpp_code git:(main) ✗
```

Experiments : 9 Graph Traversal Techniques

Aim:
Performing Breadth First Search (BFS) traversal on Graph data structure.
Objective:
Writing C++ program to perform BFS traversal on Graph.

Program:

```cpp
#include<iostream>
#include<list>
#include<map>
#include<queue>

using namespace std;

map<int, list<int>> adj;

void addEdge(int s, int e) {
    if(adj.find(s) == adj.end()) adj.insert(make_pair(s, list<int>()));
    if(adj.find(e) == adj.end()) adj.insert(make_pair(e, list<int>()));
    adj.at(s).push_back(e);
    adj.at(e).push_back(s);
}

void dfs(int n, bool* visited) {
    if(visited[n]) return;
    visited[n] = true;
    cout<< n << " ";
    list<int> l = adj.at(n);
```

```
    list<int>::iterator it;
    for(it = l.begin(); it != l.end(); it++) {
        dfs(*it, visited);
    }
}

void bfs(int n, int size) {
    bool* visited = new bool[size];
    queue<int> q;
    q.push(n);
    visited[n] = true;
    while(!q.empty()) {
        list<int> l = adj.at(q.front());
        cout<< q.front() << " ";
        q.pop();
        list<int>::iterator it;
        for(it = l.begin(); it != l.end(); it++) {
            if(!visited[*it]) {
                q.push(*it);
                visited[*it] = true;
            }
        }
    }
    cout<<endl;
}

int main() {
    // no of nodes
    int n;
    cout<< "Enter no of nodes : ";
    cin >>n;

    // no of edges
    int edges;
    cout<< "Enter no of edges : ";
    cin >>edges;
    cout<<"Enter space seperated nodes : "<< endl;
    for(int i=0;i< edges; i++) {
        int s, e;
        cin>>s;
        cin>>e;
        addEdge(s, e);
    }

    int start;
    cout<<"Enter the start node : ";
    cin >> start;

    cout<< "dfs : ";
    dfs(start, new bool[n]);
    cout<<"\nbfs : ";
    bfs(start, n);
    return 0;
}
```

```
➜  cpp_code git:(main) ✗ g++ p9/Graph.cpp && ./a.out
Enter no of nodes : 5
Enter no of edges : 4
Enter space seperated nodes :
1 0
2 0
4 0
2 3
Enter the start node : 2
dfs : 2 0 1 4 3
bfs : 2 0 3 1 4
➜  cpp_code git:(main) ✗
```

Experiments : 10 Minimum Spanning Tree

Aim: Finding Minimum Spanning tree using Kruskal's algorithm.

Objective:
Writing c++ program to find minimum spanning tree using Kruskal's algorithm from a given graph.

Program :

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

struct Edge
{
    int src, dest, weight;
};

struct Graph
{
    int V, E;
    vector<Edge> edges;
};

bool compare(Edge e1, Edge e2)
{
    return e1.weight < e2.weight;
}

int find(int parent[], int i)
{
    if (parent[i] == i)
        return i;
    return find(parent, parent[i]);
}

void Union(int parent[], int rank[], int x, int y)
{
    int xroot = find(parent, x);
    int yroot = find(parent, y);
    if (rank[xroot] < rank[yroot])
        parent[xroot] = yroot;
    else if (rank[xroot] > rank[yroot])
        parent[yroot] = xroot;
    else
    {
        parent[yroot] = xroot;
        rank[xroot]++;
    }
}

void kruskalMST(Graph graph)
```

```
{
    int V = graph.V;
    vector<Edge> result;
    int e = 0, i = 0;

    sort(graph.edges.begin(), graph.edges.end(), compare);

    int parent[V];
    int rank[V];

    for (int v = 0; v < V; v++)
    {
        parent[v] = v;
        rank[v] = 0;
    }

    while (e < V - 1 && i < graph.E)
    {
        Edge next_edge = graph.edges[i++];
        int x = find(parent, next_edge.src);
        int y = find(parent, next_edge.dest);
        if (x != y) {
            result.push_back(next_edge);
            Union(parent, rank, x, y);
            e++;
        }
    }

    cout << "Minimum spanning tree of the given graph:" << endl;
    for (i = 0; i < result.size(); i++){
        cout << result[i].src << " -- " << result[i].dest << " with weight " <<
result[i].weight << endl;
    }
}

int main()
{
    Graph graph = {5, 7, {{0, 1, 2}, {1, 2, 3}, {2, 3, 1}, {3, 0, 4}, {0, 4, 1}, {1, 4, 3},
{2, 4, 5}}};
    kruskalMST(graph);
    return 0;
}
```

```
ak@DESKTOP-R11F3IS:~/Data-Structure-in-cpp$ g++ p10/Kruskal.cpp && ./a.out
Minimum spanning tree of the given graph:
2 -- 3 with weight 1
0 -- 4 with weight 1
0 -- 1 with weight 2
1 -- 2 with weight 3
ak@DESKTOP-R11F3IS:~/Data-Structure-in-cpp$
```

Project in C++

LRU Cache

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

class Node {
public:
    int key;
    string value;
    Node* prev;
    Node* next;
    Node(int k, string v): key(k), value(v), prev(NULL), next(NULL) {}
};

class LRUCache {
private:
    int capacity;
    unordered_map<int, Node*> map;
    Node* head;
    Node* tail;

public:
    LRUCache(int cap) {
        capacity = cap;
        head = NULL;
        tail = NULL;
    }
    void moveToHead(Node* node) {
        if(head == NULL) {
            head = node;
            tail = node;
        }
        if (node == head) {
            return;
        } else if (node == tail) {
            tail = node->prev;
            tail->next = NULL;
        } else {
            if(node->prev != NULL) node->prev->next = node->next;
            if(node->next != NULL) node->next->prev = node->prev;
        }
        node->next = head;
        node->prev = NULL;
        if (head != NULL) head->prev = node;
        head = node;
        if (tail == NULL) tail = head;
    }
    void removeTail() {
        if (tail == NULL) {
            return;
        }
        map.erase(tail->key);
        if (tail == head) {
```

```
            delete tail;
            head = NULL;
            tail = NULL;
        } else {
            tail = tail->prev;
            delete tail->next;
            tail->next = NULL;
        }
    }
    string get(int key) {
        if (map.find(key) == map.end()) {
            return NULL;
        }
        Node* node = map[key];
        moveToHead(node);
        return node->value;
    }
    void print() {
        Node* temp = head;
        while(temp != NULL) {
            cout<< temp->value<< " ";
            temp = temp->next;
        }

        cout<< endl;
    }
    void put(int key, string value) {
        if (map.find(key) != map.end()) {
            Node* node = map[key];
            node->value = value;
            moveToHead(node);
        } else {
            Node* node = new Node(key, value);
            map[key] = node;
            if (map.size() > capacity) {
                removeTail();
            }
            moveToHead(node);
        }
    }
    void printMenu() {
        cout<<"~~~~~~~ Menu ~~~~~~~"<< endl;
        cout<<"1. add data"<< endl;
        cout<<"2. get data"<< endl;
        cout<<"7. view"<< endl;
        cout<<"8. clear screen"<< endl;
        cout<<"9. exit"<< endl;
    }
};
int main() {
    int capacity;
    cout<<"Enter the capacity : ";
    cin >> capacity;

    LRUCache cache(capacity);
    cache.printMenu();
```

```
    int key;
    string value;
    while(1) {
        cout<<"Enter your choice: ";
        char choice;
        cin >> choice;
        switch (choice){
            case '1':
                cout<<"Enter the key and value to add : ";
                cin >> key;
                cin >> value;
                cache.put(key, value);
                break;
            case '2':
                cout<<"Enter the key to fetch : ";
                cin >> key;
                cout<< cache.get(key) << endl;
                break;
            case '7':
                cache.print();
                break;
            case '8':
                system("clear"); // clear the screen
                cache.printMenu();
                break;
            case '9':
                exit(0);
                break;
            default:
                cout<<"Invalid Choice"<< endl;
                break;
        }
    }
    return 0;
}
```

```
ak@DESKTOP-R11F3IS:~/Data-Structure-in-cpp$ g++ project/LRUCache.cpp && ./a.out
Enter the capacity : 3
~~~~~~~~ Menu ~~~~~~~~
1. add data
2. get data
7. view
8. clear screen
9. exit
Enter your choice: 1
Enter the key and value to add : 1 java
Enter your choice: 1
Enter the key and value to add : 2 c++
Enter your choice: 1
Enter the key and value to add : 3 golang
Enter your choice: 7
golang c++ java
Enter your choice: 2
Enter the key to fetch : 2
c++
Enter your choice: 7
c++ golang java
Enter your choice: 2
Enter the key to fetch : 1
java
Enter your choice: 7
java c++ golang
Enter your choice: 1
Enter the key and value to add : 4 rust
Enter your choice: 7
rust java c++
Enter your choice: 1
Enter the key and value to add : 5 kotlin
Enter your choice: 7
kotlin rust java
Enter your choice: 9
ak@DESKTOP-R11F3IS:~/Data-Structure-in-cpp$
```