SIX WEEKS SUMMER TRAINING REPORT

on

Solving competitive problems using data structures.

Submitted by

Ritik Katoch

Registration No: 12106091

Program Name: B.Tech.(CSE)

Under the Guidance of

Mr. Ravi Kant Sahu

Assistant Professor

School of Computer Science and Engineering
Lovely Professional University. Phagwara
(June- July, 2023)

DECLARATION

I hereby declare that I have completed my six weeks summer training at **Human Resource Development Center, Lovely Professional University, Phagwara (Punjab)** from **6th June 2023** to **17th July 2023** under the guidance of **Mr. Ravi Kant Sahu**. I declare that I have worked with full dedication during these six weeks of training and my learning outcomes fulfill the requirements of training for the award of degree of **B.Tech.(Computer Science & Engineering)** at Lovely Professional University, Phagwara.

Ritik Katoch
Registration No: 12106091

ACKNOWLEDGEMENT

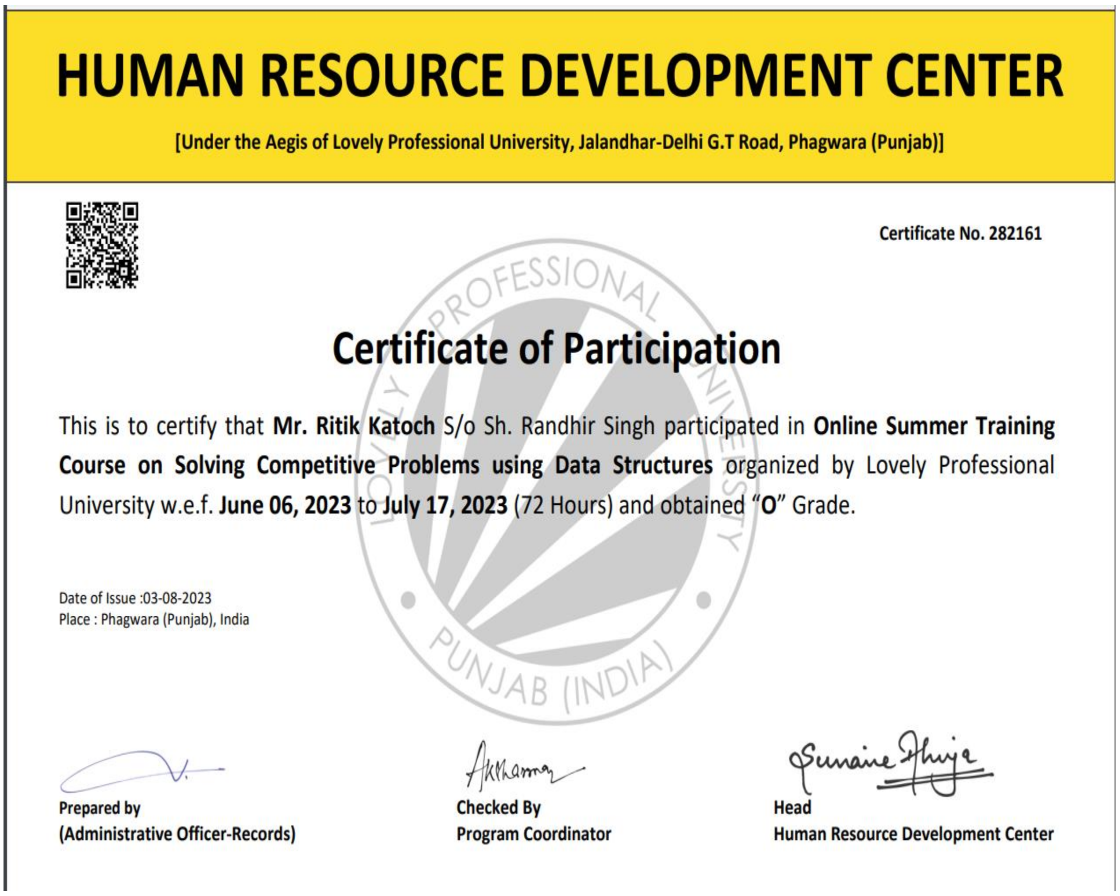
To acknowledge all the persons who had helped in completing the training and project is not possible for any trainee. However, despite all that, it becomes a foremost responsibility of the trainee and the part of research ethics toacknowledge those who had played a significant role in the completion of the training and project.

So, in the same sequence at very first, we would like to acknowledge Mr. Ravi Kant Sahu, the mentor of our Training. He has been instrumental in guiding and helping us while undergoing the planning phase of our project and helped clear the concepts related to the topics and project. Because the Training and Placement Cell of School of Computer Science & Engineering has allowed me to do the summer training, I would like to thank for the same. I would like to thank Human Resource Development Center, LPU also for organizing such a wonderful summer training program.

Later, I would like to confer the flower of acknowledgment to our parents because of whom we got the existence in the world for the inception and the conception of this training and project. Rest all those people who helped us are not only a matter of acknowledgment but also authorized to share our success.

With Regards

CERTIFICATE



HUMAN RESOURCE DEVELOPMENT CENTER

[Under the Aegis of Lovely Professional University, Jalandhar-Delhi G.T Road, Phagwara (Punjab)]

Certificate No. 282161

Certificate of Participation

This is to certify that **Mr. Ritik Katoch** S/o Sh. Randhir Singh participated in **Online Summer Training Course on Solving Competitive Problems using Data Structures** organized by Lovely Professional University w.e.f. **June 06, 2023** to **July 17, 2023** (72 Hours) and obtained "**O**" Grade.

Date of Issue :03-08-2023
Place : Phagwara (Punjab), India

[illegible] [illegible] [illegible]

Prepared by
(Administrative Officer-Records)

Checked By
Program Coordinator

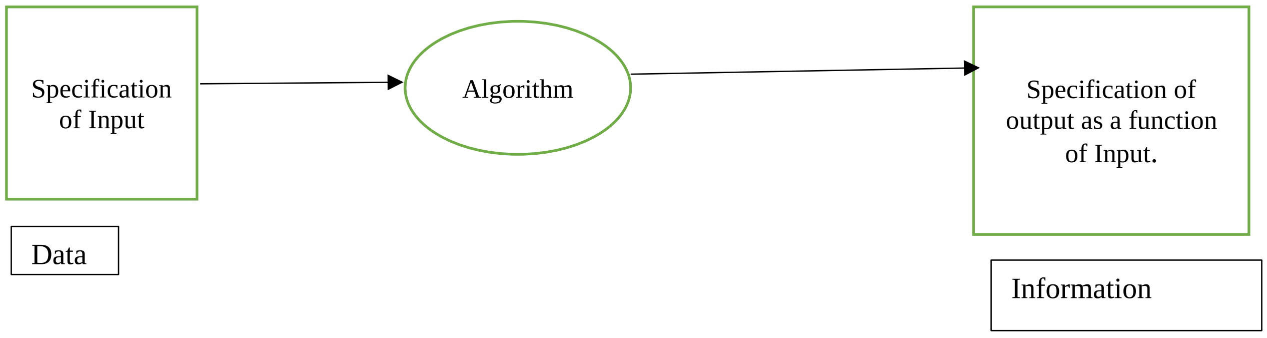
Head
Human Resource Development Center

TABLE OF CONTENTS

INTRODUCTION

Data Structure: A data structure is a group of data elements that provides the easiest way to store and perform different actions on the data of the computer. Logical or mathematical model of a particular orgranisation of data is known as Data structure. Data Structure works on primary memory.

Algorithm: It is a finite set of instructions.



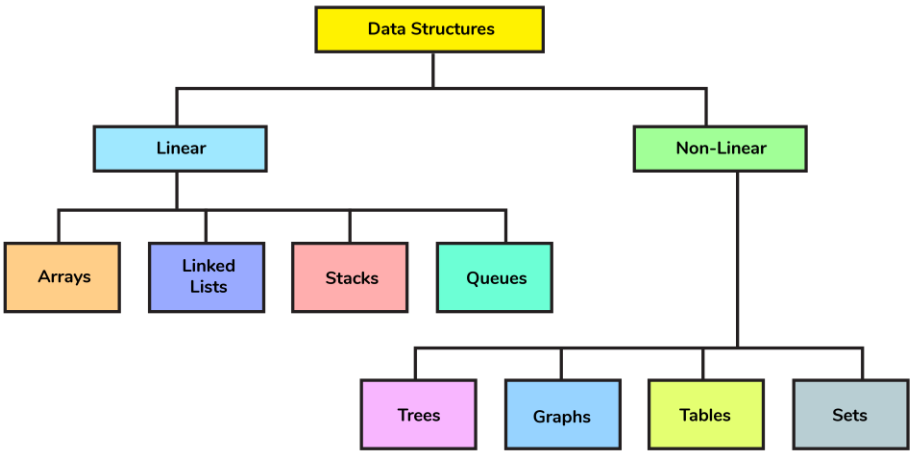
The choice of a good data structure makes it possible to perform a variety of critical operations effectively. An efficient data structure also uses minimum memory space and execution time to process the structure. A data structure is not only used for organizing the data. It is also used for processing, retrieving, and storing data. There are different basic and advanced types of data structures that are used in almost every program or software system that has been developed. So we must have good knowledge of data structures.

Technology Learnt

During the 6-week online summer training "Solving Competitive Problems using Data Structures", I have learnt basics of Data Structures and applied those concepts in solving some standard competitive coding problems using C++. The brief outlines of the contents covered is as following:

Data Structures: Data structures are a specific way of organizing data in a specialized format on a computer so that the information can be organized, processed, stored, and retrieved quickly and effectively. They are a means of handling information, rendering the data for easy use.

- **Classification of Data Structures**



Classification of Data Structures

Sorting Techniques: Sorting algorithms are used to arrange the elements of a data structure in a particular order such as ascending or descending order.

i. Bubble Sort:

```
#include<iostream>
using namespace std;
int main(){
    int arr[100],n,temp;
    cout<<"enter the size of array"<<endl;
    cin>>n;
    cout<<"enter the elements of araay"<<endl;
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }
    for(int i=0;i<n-1;i++){
        for(int j=0;j<n-i-1;j++){
            if(arr[j]>arr[j+1]){
                temp=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp;
            }
        }
    }
    cout<<"elements after sorting"<<endl;
    for(int i=0;i<n;i++){
        cout<<arr[i]<<" ";
    }
}
```

ii. Selection sort:

```
void selectionsort(int arr[],int n){
    for(int i=0;i<n-1;i++){
        int min=i;
        int j=i+1;
        for(int j=i+1;j<n;j++){
            if(arr[j]<arr[min]){
                min=j;
            }
        }
        if(min!=j){
            int temp=arr[i];
            arr[i]=arr[min];
            arr[min]=temp;

        }
    }
```

iii Insertion sort:

```
void insertionsort(int arr[],int n){
    int i;
    for(i=1;i<n;i++){
        int temp=arr[i];
        int j=i-1;
        while(j>=0 && temp<arr[j]){
            arr[j+1]=arr[j];
            j--;
        }
        arr[j+1]=temp;
    }
}
```

Sorting	swaps	Space	O(BIG-O)(worst case)	Theta(average case)	Omega(best case)
Bubble	n^2	1	n^2	n^2	n^2
Selection	n	1	n^2	n^2	n^2
Insertion	0	1	n^2	n^2	n

Comparing different sorting techniques on basis of time and space complexities.

```
}
```

Searching techniques: Searching is used to find the location of particular element or the data present in the array or any other data structure.

Binary search:

```
#include <iostream>

using namespace std;

int size; //to store the size of array
int *arr; //array reference variable
int n; //Number of elements in the array
int binarysearch(int key){
   int beg=0;
   int end=n-1;
   while(beg<=end){
      int mid=(beg+end)/2;
      if(key==arr[mid]){
         return mid;
      }
      else if(key<arr[mid]){
         end=mid-1;
      }
      else{
         beg=mid+1;
      }
   }
   return -1;
}
```

```
void printArray()
{
//Traversing the array
for(int i=0; i<n; i++)
cout<<arr[i] <<"\t";
}

int main()
{
cout<<"Enter the size of Array";
cin>> size;
arr = new int[size];

cout<<"How many elements you want to enter";
cin>>n;

for(int i=0; i<n; i++)
{
cin>>arr[i]; //storing the elements in the array
}
printArray();
cout<<endl<<"enter the key you want to search";
int k;
cin>>k;
int r=binarysearch(k);
if(r==-1){
   cout<<endl<<k<<" "<<"is not present in the array";
}
else{
   cout<<endl<<k<<" "<<"is present at the index"<<" "<<r;
}
return 0;
}
```

SNO.	BEST CASE	AVERAGE CASE	WORST CASE
1. Linear	1	n	n
2. Binary	1	Logn	Logn

DATA STRUCTURES:

1. **Arrays:**

   It is the linear data structure which has following properties:

   1. Fixed size
   2. Homogeneous elements
   3. Indexed access
   4. Contiguous memory allocation

   Insertion in array at a given index K.

   ```
   for(int i=n;i>k;i--)//shifting logic

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

Problem statements based on array:

1.Search an element in an rotated_sorted array:

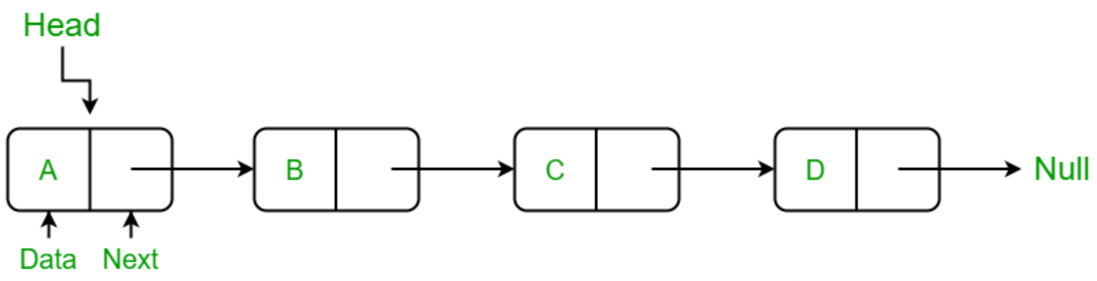
```
class Solution {
public:
   int search(vector<int>& nums, int target) {
      int n=nums.size();
      int low=0;
      int high=n-1;
      while(low<=high){
         int mid=(low+high)/2;
         if(nums[mid]==target){
            return mid;
         }
         else if(nums[low]<=nums[mid]){
            if(nums[low]<=target && target<=nums[mid])
            high=mid-1;
            else
            low=mid+1;
         }
         else{
            if(target>=nums[mid]&& target<=nums[high])
            low=mid+1;
            else
            high=mid-1;
         }
      }
      return -1;

   }
};
```

2. **Sort an array of 0s,1s and 2s|Dutch National Flag problem:**

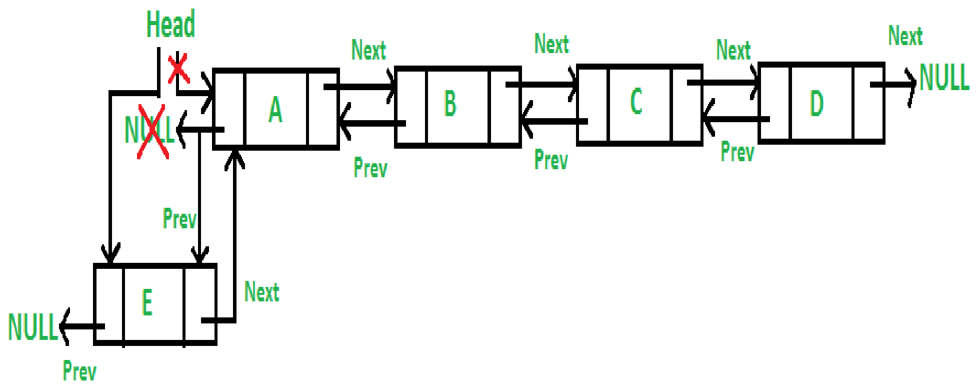
```
#include <iostream>//only for binary numbers dutch flag algorithm
#include <bits/stdc++.h>

using namespace std;
void sort(int a[],int k){
    int l=0;
    int h=k-1;
    int mid=0;
    while(mid<=h){
        switch(a[mid]){
            case 0:
            swap(a[l++],a[mid++]);
            break;
            case 1:
            mid++;
            break;
            case 2:
            swap(a[mid],a[h--]);
            break;

        }
    }
}

void print(int a[],int k){
    cout<<"sorted array is "<<endl;
    for(int i=0;i<k;i++){
        cout<<a[i]<<" ";
    }
}

int main()
{
    int arr[100],n;
    cout<<"enter the size of array"<<endl;
    cin>>n;
    cout<<"enter the elements of array"<<endl;
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }
    sort(arr,n);
    print(arr,n);
return 0;
  }
```

2. Linked List:

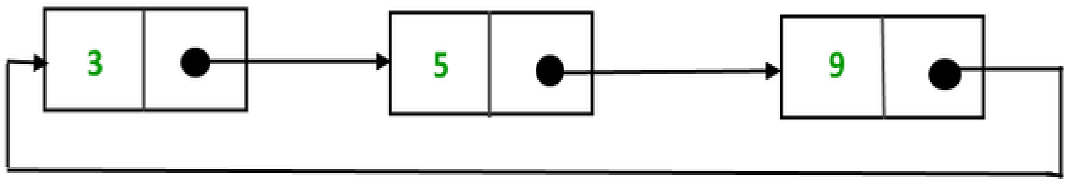
1. Singly linked list: A singly linked list is a linear data structure in which the elements are not stored in contiguous memory locations and each element is connected only to its next element using a pointer.



2. Doubly Linked list:



3. Circular Linked list:



Problem statements based on Linked list:

1. Detect a loop in linked list:

```
/*

struct Node
{
    int data;
    struct Node *next;
    Node(int x) {
        data = x;
        next = NULL;
    }

*/
class Solution
{
    public:
```

```
    //Function to check if the linked list has a loop.
    bool detectLoop(Node* head)
    {
       Node *fast=head->next;
       Node *slow=head;
       while(slow!=fast){
           if(fast==NULL || fast->next==NULL){
               return false;
           }
           slow=slow->next;
           fast=fast->next->next;
       }
           return true;
       }

};
```

2. Find middle element in a linked list:

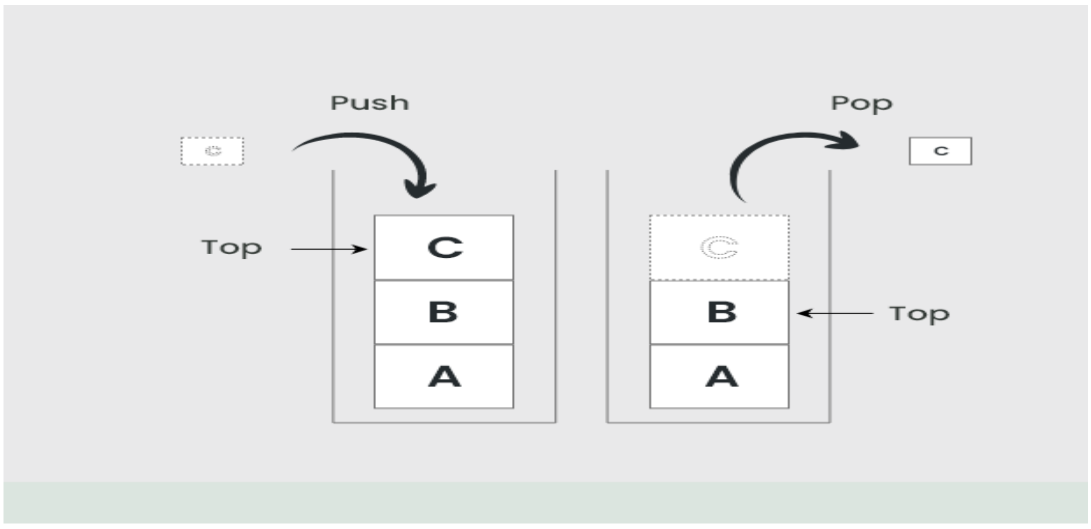
```
/* Link list Node
struct Node {
    int data;
    Node* next;

    Node(int x){
        data = x;
        next = NULL;
    }

}; */
class Solution{
    public:
    /* Should return data of middle node. If linked list is empty, then  -1*/
    int getMiddle(Node *head)
    {
        Node *slow=head;
        Node *fast=head;
        if(head==NULL)
        return -1;

   while( fast!= NULL  &&  fast->next != NULL )
    {
        fast=fast->next->next;

        slow=slow->next;
    }
    return slow->data;
}
};
```

3**. Find the length of the loop:**

```
/*

struct Node {
    int data;
    struct Node *next;
    Node(int x) {
        data = x;
        next = NULL;
    }
};

*/

//Function to find the length of a loop in the linked list.
int countNodesinLoop(struct Node *head)
{
   Node *slow=head;
   Node *fast=head;

while(fast && fast->next){
    slow=slow->next;
    fast=fast->next->next;
    if(slow==fast){
        int c=1;
        Node *a=slow->next;
        while(a!=slow){
            c++;
            a=a->next;

        }
        return c;
    }
}
return 0;

}
```

3. STACK:

Stack is a linear data structure that follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out). LIFO implies that the element that is inserted last, comes out first and FILO implies that the element that is inserted first, comes out last.



Problems statements based on stack :

1. Reverse a string using stack:

```
char* reverse(char *S, int len)
{
   //code here
   int i=0;
   stack<char>s;
   while(i<len){
      s.push(S[i]);
      i++;
   }
   i=0;
   while(i<len){
      S[i]=s.top();
      s.pop();
      i++;
   }
   return S;
     }
```

2. Implement two stacks in an array:

```
class twoStacks
{
   int *arr;
   int size;
   int top1, top2;
   public:

   twoStacks(int n=100)
   {
      size = n;
```

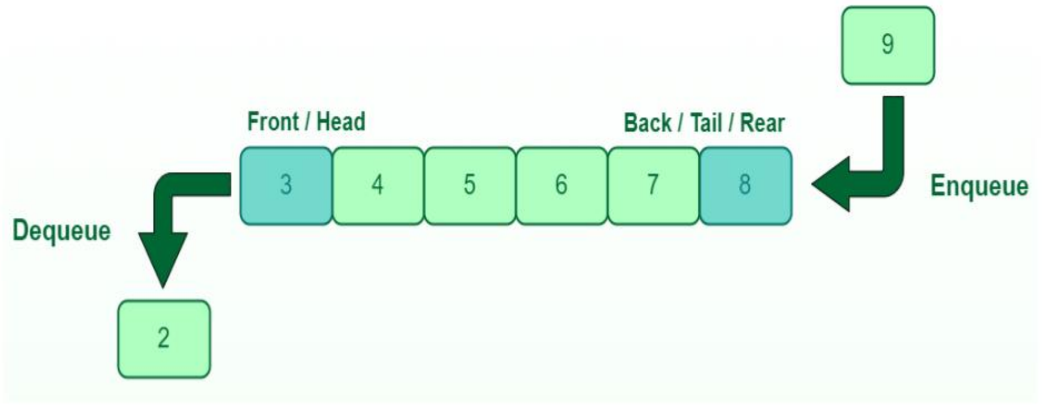
```
        arr = new int[n];
        top1 = -1;
        top2 = size;
    }

    //Function to push an integer into the stack1.
    void push1(int x){
    if(top1<top2-1){
        top1++;
        arr[top1]=x;
    }



          }
```

3. **Parenthesis checker:**

```
class Solution
{
    public:
    //Function to check if brackets are balanced or not.
    bool ispar(string x)
    {

        stack<char> s;
        for (char ch : x)
        {
            if (ch == '(' || ch == '[' || ch == '{')
            {
                s.push(ch);
            }
            else if (s.empty())
            {
                return false;
            }
            else
            {
                char c = s.top();
                s.pop();
                if ((ch == ')' && c != '(') || (ch == ']' && c != '[') || (ch == '}' && c != '{'))
                {
                   return false;
                }

             }
          }
          return s.empty();
       }

    };
```

4. QUEUES:

A Queue is defined as a linear data structure that is open at both ends and the operations are performed in First In First Out (FIFO) order.



Queue Data Structure

Queue using two stacks:

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

class MyQueue
{
    stack<int> s1;
    stack<int> s2;
    public:
     void myPush(int i)
     {
         s1.push(i);
     }

    int myPop()
    {
        while(!s1.empty())
        {
             int t=s1.top();
             s1.pop();
            //int t=s1.pop();
             s2.push(t);
        }
        int r=s2.top();
          s2.pop();
        while(!s2.empty())
        {
            int t=s2.top();
            s2.pop();
            s1.push(t);
        }
        return r;
    }
};

int main()
{
    MyQueue  m;
    m.myPush(5);
    m.myPush(10);
    m.myPush(15);
    cout<<m.myPop()<<endl;
    return 0;
}
```

Queues various operations:

```
#include <iostream>
using namespace std;
#define MAX 5

class MyQueue
{
    int q[MAX];
    int front, rear;
    public:
     MyQueue()
     {
         front=-1;
         rear=-1;
     }
    bool isEmpty();
    bool isFull();
    void enQueue(int);
    void deQueue();
    void peek();
    void display();

};
void MyQueue::display()
{
     if(front==-1 && rear==-1)
        cout<<"Q is underflow"<<endl;
    else
    {
        for(int i=front;i<=rear;i++)
            cout<<q[i]<<endl;
     }
}
void MyQueue::peek()
{
    if(front==-1 && rear==-1)
        cout<<"Q is underflow"<<endl;
    else
```

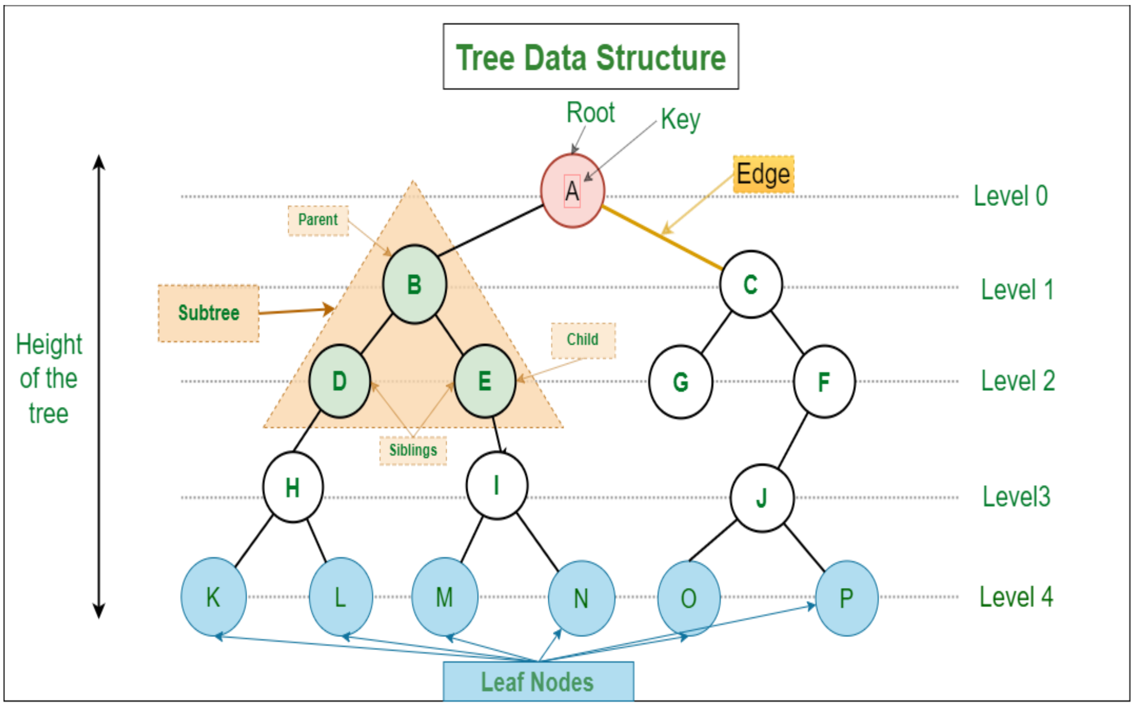
```
        cout<<"Front of q:"<<q[front]<<endl;
}

void MyQueue::deQueue()
{
    if(front==-1 && rear==-1)
       cout<<"Queue is underflow"<<endl;
    else if(front==rear)
    {
        cout<<q[front]<<endl;
        front=-1;
        rear=-1;
    }
    else
    {
        cout<<q[front]<<endl;
        front++;
    }
}
void MyQueue::enQueue(int data)
{
    //MyQueue t;
    if(rear==MAX-1) // if(!isFull())
      cout<<"Queue is Overflow"<<endl;
    else if(front==-1 && rear==-1)
     {
         front=0;
         rear=0;
         q[rear]=data;
     }
     else
     {
         rear++;
         q[rear]=data;

         //q[++rear]=data;
     }
}
bool MyQueue ::isEmpty()
{
    if(front==rear )
         return true;
     else
         return false;
```

```
}

bool MyQueue::isFull()
{
   if (rear==MAX-1)
      return true;
   else
      return false;
}


int main()
{
   MyQueue mq;
   cout<<mq.isEmpty()<<endl;
   cout<<mq.isFull()<<endl;
   mq.enQueue(7);
   mq.enQueue(10);
   mq.enQueue(15);
   mq.deQueue();
   mq.peek();
   mq.deQueue();
   mq.deQueue();
   mq.deQueue();
   mq.enQueue(30);
   mq.enQueue(40);
   mq.enQueue(50);
   mq.enQueue(60);
   cout<<"After insertion 6 data"<<endl;
   cout<<mq.isEmpty()<<endl;
   cout<<mq.isFull()<<endl;
   cout<<"My Queue element :"<<endl;
   mq.display();

   return 0;
}
```

5. TREES:

A tree data structure is a hierarchical structure that is used to represent and organize data in a way that is easy to navigate and search. It is a collection of nodes that are connected by edges and has a hierarchical relationship between the nodes.

The topmost node of the tree is called the root, and the nodes below it are called the child nodes. Each node can have multiple child nodes, and these child nodes can also have their own child nodes, forming a recursive structure.



Types of trees:

1.Binary tree: In a binary tree, each node can have a maximum of two children linked to it. Some common types of binary trees include full binary trees, complete binary trees, balanced binary trees, and degenerate or pathological binary trees

2.Ternary tree: A Ternary Tree is a tree data structure in which each node has at most three child nodes, usually distinguished as “left”, “mid” and “right”.

3.N-ary tree or Generic tree: Generic trees are a collection of nodes where each node is a data structure that consists of records and a list of references to its children(duplicate references are not allowed). Unlike the linked list, each node stores the address of multiple nodes**.**

Problem statements based on trees:

1.Height of a binary tree

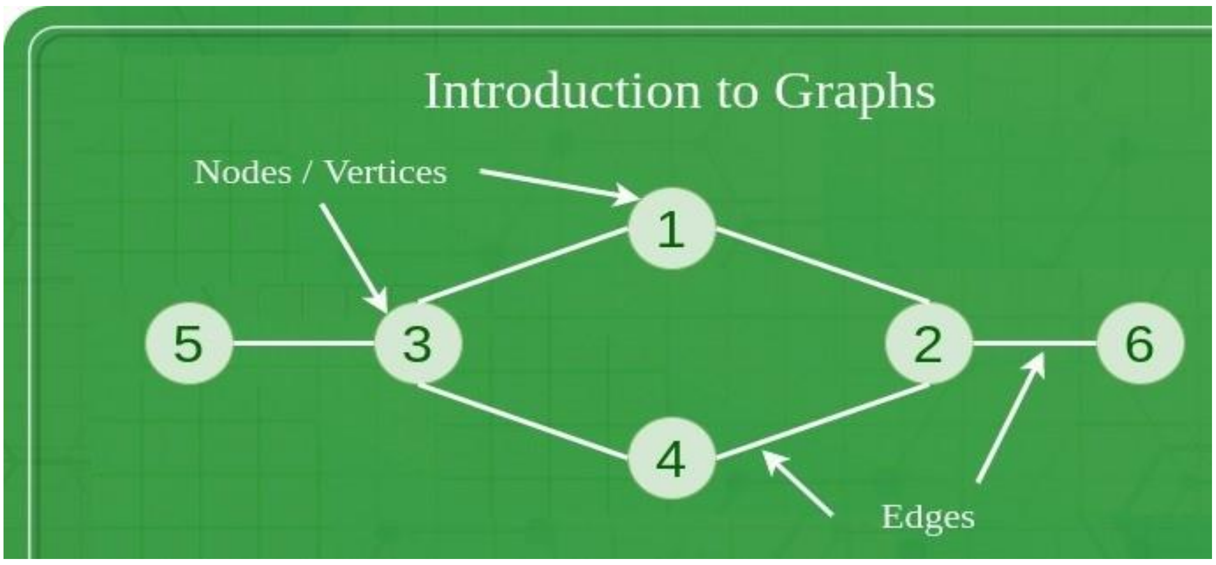
```
/*
struct Node
{
    int data;
    struct Node* left;
    struct Node* right;

    Node(int x){
        data = x;
        left = right = NULL;
    }
};
*/
class Solution{
    public:
    //Function to find the height of a binary tree.
    int height(struct Node* root){
        if(root==NULL){
            return 0;
        }
        int n1=height(root->left);
        int n2=height(root->right);
        return (n1>n2? n1+1:n2+1);

    }
```

2. Diameter of a binary tree:

```
struct Node
{
    int data;
    struct Node* left;
    struct Node* right;

    Node(int x){
        data = x;
        left = right = NULL;
    }
};
class Solution {
    private:
    int height(struct Node *node){
        if(node==NULL){
            return 0;
        }
        int left=height(node->left);
        int right=height(node->right);
        int a=max(left,right)+1;
        return a;
    }

public:
    // Function to return the diameter of a Binary Tree.
    int diameter(Node* root) {
        if(root==NULL){
            return 0;
        }
        int left=diameter(root->left);
        int right=diameter(root->right);
        int y=height(root->left)+height(root->right)+1;
        int a=max(max(left,right),y);
        return a;
    }
};
```

6.Graphs:

A Graph is a non-linear data structure consisting of vertices and edges. The vertices are sometimes also referred to as nodes and the edges are lines or arcs that connect any two nodes in the graph. More formally a Graph is composed of a set of vertices(V) and a set of edges(E). The graph is denoted by G(E, V).



BFS of a graph:

```
class Solution {
 public:
  // Function to return Breadth First Traversal of given graph.
  vector<int> bfsOfGraph(int V, vector<int> adj[]) {
    vector<int>res;
    queue<int>q;
    q.push(0);
    map<int,int>vis;
    vis[0]=1;
    while(!q.empty()){
      int topp=q.front();
      q.pop();
      res.push_back(topp);
      for(auto child:adj[topp]){
        if(!vis[child]){
          vis[child]=1;
          q.push(child);
        }
      }
    }
    return res;
  }

};
```

Minimum spanning tree using prims algorithm:

```
#include <bits/stdc++.h>
using namespace std;
#define VRT 5
int minKey(int key[], bool mstSet[])
{
        int min = INT_MAX, min_index;
        for (int v = 0; v <VRT; v++)
                if (mstSet[v] == false && key[v] < min)
                        min = key[v], min_index = v;
        return min_index;
}
void printMST(int parent[], int g[VRT][VRT])
{
        for (int i = 1; i < VRT; i++)
                cout << parent[i] << "--" << i << "  "<< g[i][parent[i]] << " \n";
}
void primMinSpanningTree(int g[VRT][VRT])
{
        int parent[VRT];
        int key[VRT];
        bool mstSet[VRT];
        for (int i = 0; i < VRT; i++)
                key[i] = INT_MAX, mstSet[i] = false;
        key[0] = 0;
        parent[0] = -1;
        for (int count = 0; count < VRT - 1; count++) {
                int u = minKey(key, mstSet);
                mstSet[u] = true;
                for (int v = 0; v < VRT; v++)
                        if (g[u][v] && mstSet[v] == false
                                && g[u][v] < key[v])
                                parent[v] = u, key[v] = g[u][v];
        }
        printMST(parent, g);
}
int main()
{
   int g[VRT][VRT]= {

                {0, 2, 0, 7, 6},
                {2, 0, 1, 0, 4},
                {0, 1, 0, 3, 2},
                {7, 0, 3, 0, 5},
                {6, 4, 2, 5, 0}
               };
     primMinSpanningTree(g);
     return 0;
}
```

REASON FOR CHOOSING THIS TECHNOLOGY

1) **Write Optimized And Scalable Code:** Once we have knowledge about data structure and algorithms, we can determine which data structure and algorithm to chose in various situations and conditions.

2) **Effective Use Of Time And Memory:** Having Knowledge about data structures and algorithm will help in writing codes that run faster and require less storage.

3) **Better Job Opportunities:** Data structure and algorithm questions and frequently asked in Job interviews of various organizations like Google, Facebook, and so on**.**

Profile of the Problem

A problem profile for a contact management system involves identifying and describing the key aspects and challenges related to developing or improving such a system.

1. Problem Statement:

The current contact management system lacks essential features and user-centric design elements, resulting in a suboptimal user experience and decreased efficiency in managing contacts. The goal is to identify and address these shortcomings to create a more intuitive, functional, and user-friendly contact management system.

2. Key Challenges:

a. Inadequate User Interface and Experience:

The existing system may have a cluttered or confusing user interface, making it challenging for users to navigate and perform tasks efficiently. This can lead to frustration and errors in contact management.

b. Limited Functionality:

The current system might lack crucial features, such as integration with popular email platforms, automatic contact syncing, tagging and categorization, customizable fields, and bulk operations. This limitation restricts users from effectively managing and organizing their contacts.

c. Data Security and Privacy:

Maintaining data security and privacy is essential in a contact management system. Ensuring that user data is stored securely and compliant with relevant regulations, such as GDPR or HIPAA, is a significant challenge.

Existing System

A Contact Management System (CMS) is a software application designed to help individuals and businesses organize, store, and manage their contact information effectively.

Here are a few examples of existing systems for contact management:

1. Microsoft Outlook: Outlook is a popular personal information manager that includes features for email, calendar, task management, and contact organization. It's widely used in both personal and business contexts.

2. Google Contacts: Google Contacts is a cloud-based contact management system that integrates with other Google services like Gmail and Google Calendar. It offers features for grouping, tagging, and organizing contacts.

3. Apple Contacts: Apple Contacts, formerly known as Address Book, is integrated into the macOS and iOS ecosystem. It allows users to manage and sync contacts across Apple devices.

4. HubSpot CRM: HubSpot CRM is a comprehensive customer relationship management platform that includes contact management features. It's suitable for businesses looking to manage their contacts, sales activities, and marketing efforts in one place.

5.Salesforce: Salesforce is a widely used CRM platform that offers robust contact management capabilities along with sales automation, customer service, and marketing functionalities for businesses of all sizes.

6.Zoho CRM: Zoho CRM is another popular CRM solution that provides contact management, sales automation, marketing automation, and other features aimed at streamlining business processes.

7.Pipedrive: Pipedrive is a CRM and sales management platform with a strong focus on pipeline management. It helps businesses organize contacts, deals, and sales activities.

Problem Analysis

1. Product definition:

A Contact Management System is a software solution designed to help individuals and businesses efficiently organize, store, and manage their contact information. This system streamlines the process of storing, updating, and retrieving contact details, enabling users to maintain strong relationships with their contacts and improve communication.

2.Feasibility Analysis:

A feasibility analysis for a contact management system involves evaluating the practicality and potential success of developing and implementing such a system. This analysis typically covers three main aspects: technical feasibility, economic feasibility, and operational feasibility.

1.Technical Feasibility:

This aspect assesses whether the proposed contact management system can be developed using the available technology and resources. Consider the following factors:

Technology: Does the required technology and infrastructure exist to build the system? Is the development platform appropriate for the system's needs?

Skills and Expertise: Do you have or can you acquire the necessary skills to develop, maintain, and support the system?

Integration: Can the system integrate with existing software, databases, or systems within the organization?

2.Economic Feasibility:

This aspect focuses on determining if the project is financially viable and if the benefits outweigh the costs. Consider the following factors:

Cost Estimation: Calculate the development, implementation, and maintenance costs. Include hardware, software, staffing, training, and any other relevant expenses.

Benefits: Identify the potential benefits of the system, such as improved efficiency, time savings, reduced errors, and enhanced customer relationships.

Return on Investment (ROI): Assess whether the expected benefits will provide a favorable ROI over a reasonable period.

3. Operational Feasibility:

Operational feasibility evaluates how well the proposed system aligns with the organization's processes and goals. Consider the following factors:

User Acceptance: Will the system be user-friendly and meet the needs of its intended users? Consider usability, user training, and change management.

Organizational Impact: Analyze how the new system will affect existing workflows, processes, and employee roles. Will it require significant changes?

Scalability: Can the system accommodate future growth in terms of data volume, user base, and functionality?

Software Requirement Analysis

1. Purpose and Scope:

Is it intended for personal use, small businesses, or larger enterprises? Identify the scope of the system, such as the features and functionalities it should include.

2. User Requirements:

Identify and categorize the different types of users who will interact with the system, such as administrators, employees, and clients. Gather their specific requirements, such as the ability to add, edit, delete, and search for contacts, manage groups or categories, and perform bulk operations.

3. Functional Requirements:

Specify the functional capabilities that the system must possess. This might include:

. User authentication and authorization

. Contact creation, editing, and deletion

. Search and filtering of contacts based on various attributes

. Grouping and categorization of contacts

. Import and export of contact data

. Integration with email or messaging platforms

. Notifications for upcoming events or follow-ups

4. Non-Functional Requirements:

These are quality attributes that define the system's behavior beyond its specific functionalities:

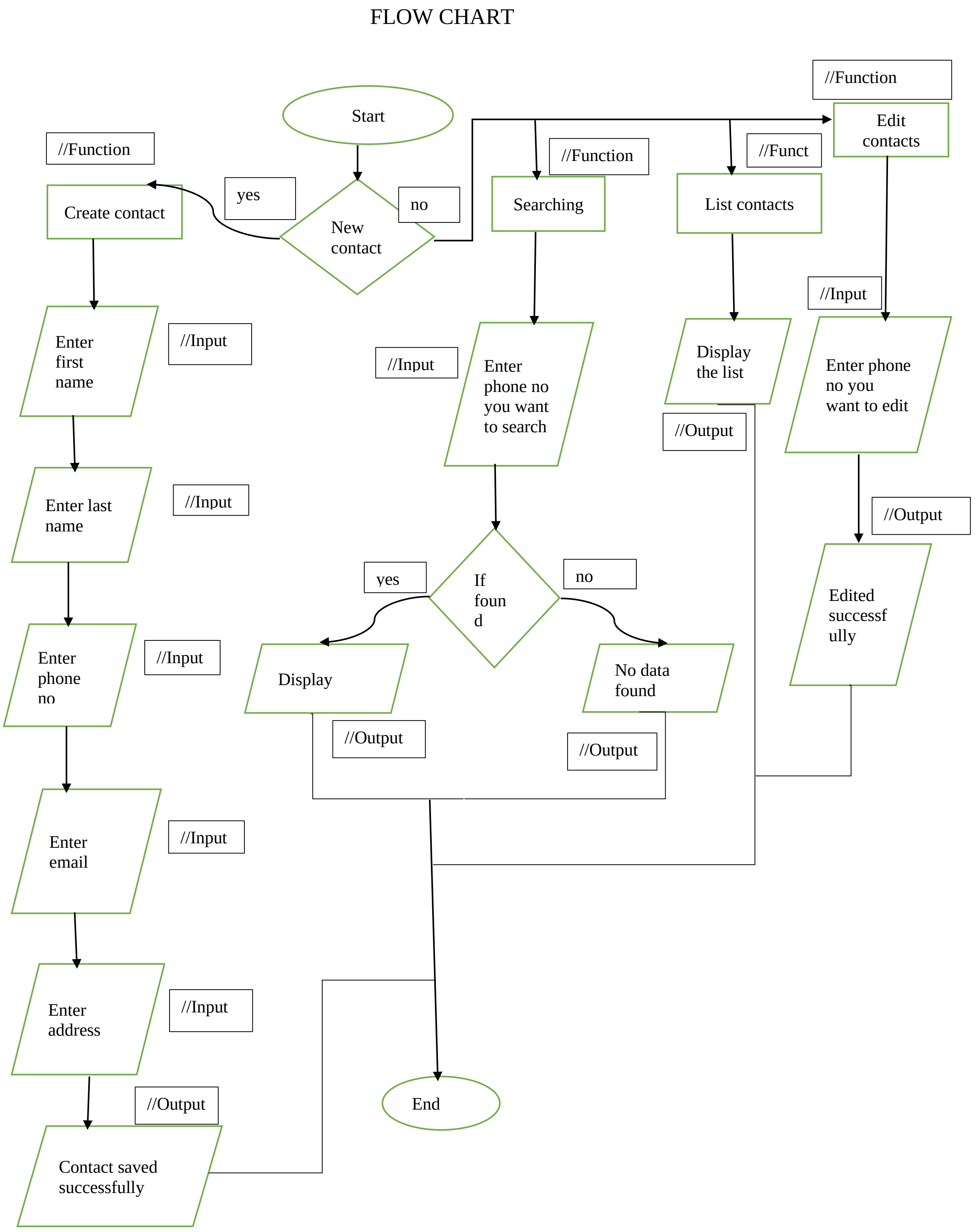
Performance: Specify response times, database query performance, and system scalability requirements.

Security: Define authentication mechanisms, access controls, and data encryption for sensitive contact information.

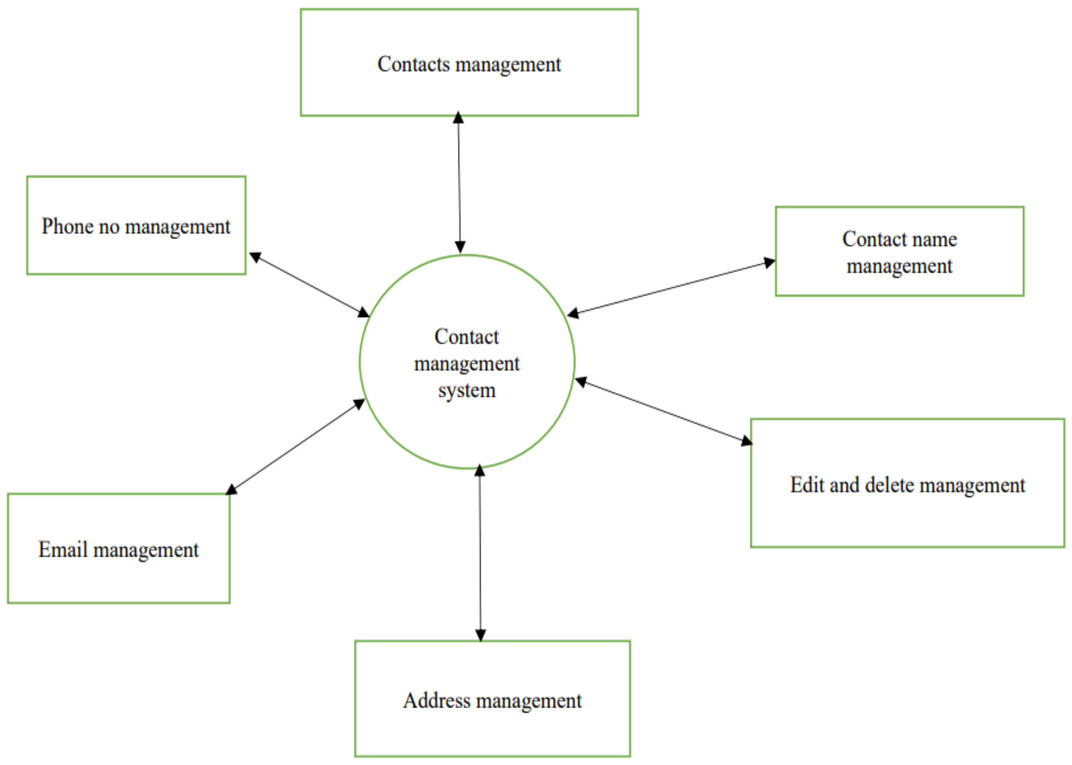
Usability: Describe the user interface's ease of use, navigation, and user experience.

Design

1. Flow chart



2.Data Flow Diagram:

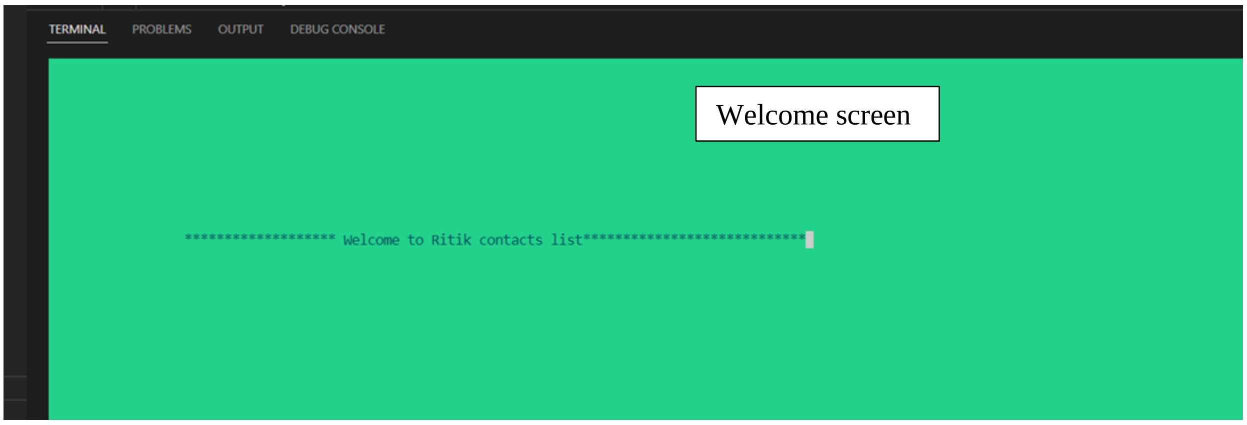


Zero level DFD

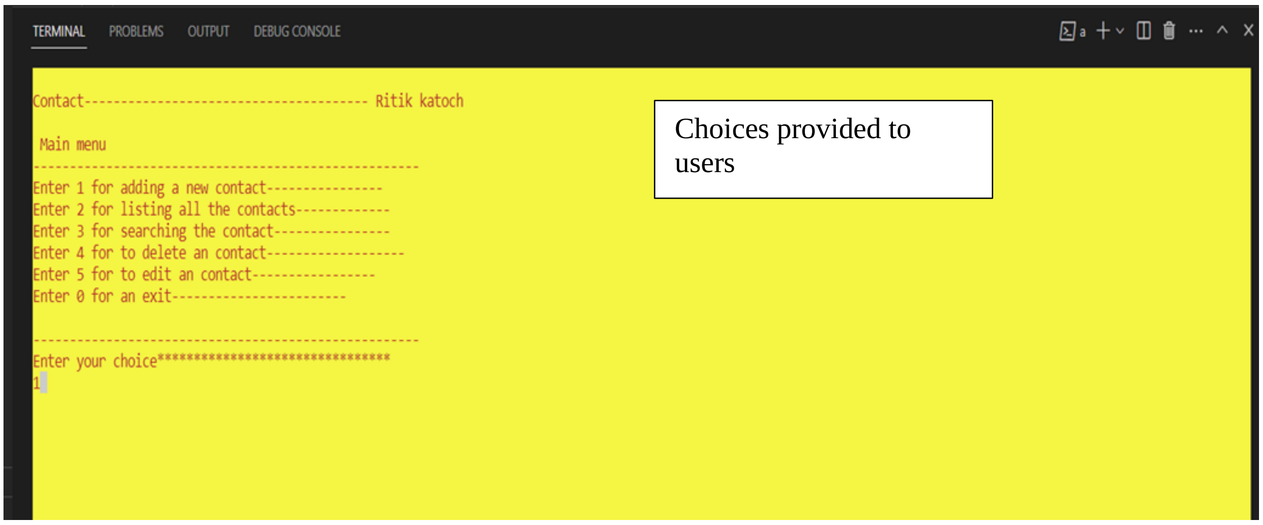
Implementation

Project code:

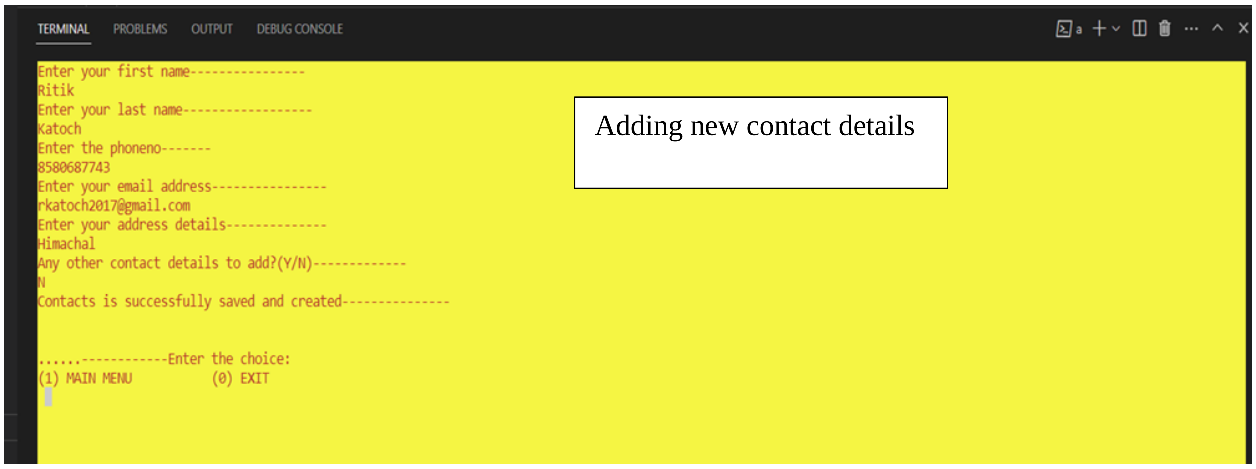
```
#include <iostream>
#include<conio.h>
#include<fstream>

using namespace std;
class contact{
 private:
    char firstname[100],lastname[100],email[100],address[100];
    long long phoneno;

 public:
   void createcontact(){
      cout<<"Enter your first name----------------"<<endl;
      cin>>firstname;

      cout<<"Enter your last name------------------"<<endl;
      cin>>lastname;

      cout<<"Enter the phoneno-------"<<endl;
      cin>>phoneno;

      cout<<"Enter your email address----------------"<<endl;
      cin>>email;

      cout<<"Enter your address details--------------"<<endl;
      cin>>address;
   }
   void contactdetails(){
      cout<<"Name---"<<firstname<<" "<<lastname<<endl;
      cout<<"Email----"<<email<<endl;
      cout<<"address-----"<<address<<endl;
      cout<<"phone number------"<<phoneno;

   }
   void writeonfile(){
      char ch;
      ofstream f1;
      f1.open("CMS.dat",ios::binary | ios::app);
      do{
         createcontact();
         f1.write(reinterpret_cast<char*>(this),sizeof(*this));
         cout<<"Any other contact details to add?(Y/N)-------------";
         cin>>ch;
      }while(ch=='Y');
     cout<<"Contacts is successfully saved and created---------------"<<endl;
      f1.close();
   }
   void readfromfile(){
      ifstream f2;
      f2.open("CMS.dat",ios::binary);
      cout<<"\n--------------------------------------------------\n";
      cout<<"List of contacts";
      cout<<"\n---------------------------------------------------\n";

      while(!f2.eof()){
         if(f2.read(reinterpret_cast<char*>(this),sizeof(*this))){
             contactdetails();
             cout<<"\n------------------------------------------\n";}}
             f2.close();

         }

   void searchcontact(){
      ifstream f3;
      long long phone;
      cout<<"Enter the phone number you want to search-------------------------"<<endl;
      cin>>phone;
      f3.open("CMS.dat",ios::binary);

      while(!f3.eof()){
         if(f3.read(reinterpret_cast<char*>(this),sizeof(*this))){
             if(phone==phoneno){
                 contactdetails();
                 return;}
             }
         }
         cout<<"\n\n No contacts found in data";
         f3.close();
      }
      void deletefromfile(){
         long long phone;
         int flag=0;
         ofstream f4;
         ifstream f5;
         f5.open("CMS.dat",ios::binary);
         f4.open("temp.dat",ios::binary);

         cout<<"Enter the phone no be deleted------------------------"<<endl;
        cin>>phone;
        while(!f5.eof()){
            if(f5.read(reinterpret_cast<char*>(this),sizeof(*this))){
                if(phoneno!=phone){
                    f4.write(reinterpret_cast<char*>(this),sizeof(*this));
                    }
                    else flag=1;
                }
            }
            f5.close();
            f4.close();
            remove("CMS.dat");
            rename("temp.dat","CMS.dat");

            flag==1?
            cout<<"\tContact successfully deleted---------":
            cout<<"\tContact not found-----------------";
        }

        void editcontact(){
            long long phone;
            fstream f6;
            cout<<"Edit contacts";
            cout<<"\n-------------------------------------\n";
            cout<<"Enter the phone no you want to edit---------------------------";
            cin>>phone;

            f6.open("CMS.dat",ios::binary|ios::in|ios::out);

            while(!f6.eof()){
                if(f6.read(reinterpret_cast<char*>(this),sizeof(*this))){
                    if(phone==phoneno){
                        cout<<"Enter new details-------------------\n";
                        createcontact();

                        int pos=-1 *sizeof(*this);
                        f6.seekp(pos,ios::cur);
                        f6.write(reinterpret_cast<char*>(this),sizeof(*this));
                        cout<<endl<<endl<<"\tContact updated-------------------------------";
                        return;
                    }
                }
            }
            cout<<"\n\n no data found";
            f6.close();
        }


};
int main(){
    system("cls");
    system("Color A3");
    cout<<"\n\n\n\n\n\n\n\n\n\n\t\t ******************** Welcome to Ritik contacts
list******************************";
   getch();

   while(1){
      contact c1;
      int choice;

      system("cls");
      system("Color E4");

      cout<<"\nContact--------------------------------------- Ritik katoch";
      cout<<"\n\n Main menu";
      cout<<"\n-----------------------------------------------------\n";
      cout<<"Enter 1 for adding a new contact----------------\n";
      cout<<"Enter 2 for listing all the contacts-------------\n";
      cout<<"Enter 3 for searching the contact----------------\n";
      cout<<"Enter 4 for to delete an contact-------------------\n";
      cout<<"Enter 5 for to edit an contact-----------------\n";
      cout<<"Enter 0 for an exit------------------------\n";
      cout<<"\n-----------------------------------------------------\n";
      cout<<"Enter your choice***********************************";
      cin>>choice;

      switch(choice){
         case 1:
         system("cls");
         c1.writeonfile();
         break;

         case 2:
         system("cls");
         c1.readfromfile();
         break;

         case 3:
         system("cls");
         c1.searchcontact();
         break;

         case 4:
         system("cls");
         c1.deletefromfile();
         break;

         case 5:
        system("cls");
        c1.editcontact();
        break;
        case 0:
         system("cls");
                                cout<<"\n\n\n\n\n\t\t****************************Thank
You*************************************"<<endl;
         exit(0);
         break;

         default:
         continue;}
         int option;
         cout<<"\n\n......------------Enter the choice:\n(1) MAIN MENU\t\t(0) EXIT\n ";
         cin>>option;

         switch(option){
            case 0:
            system("cls");
                              cout<<"\n\n\n\n\n\t\t*****************************Thank
you***********************************"<<endl;
         exit(0);
         break;

          default:
          continue;}
      }
      return 0;

}
```



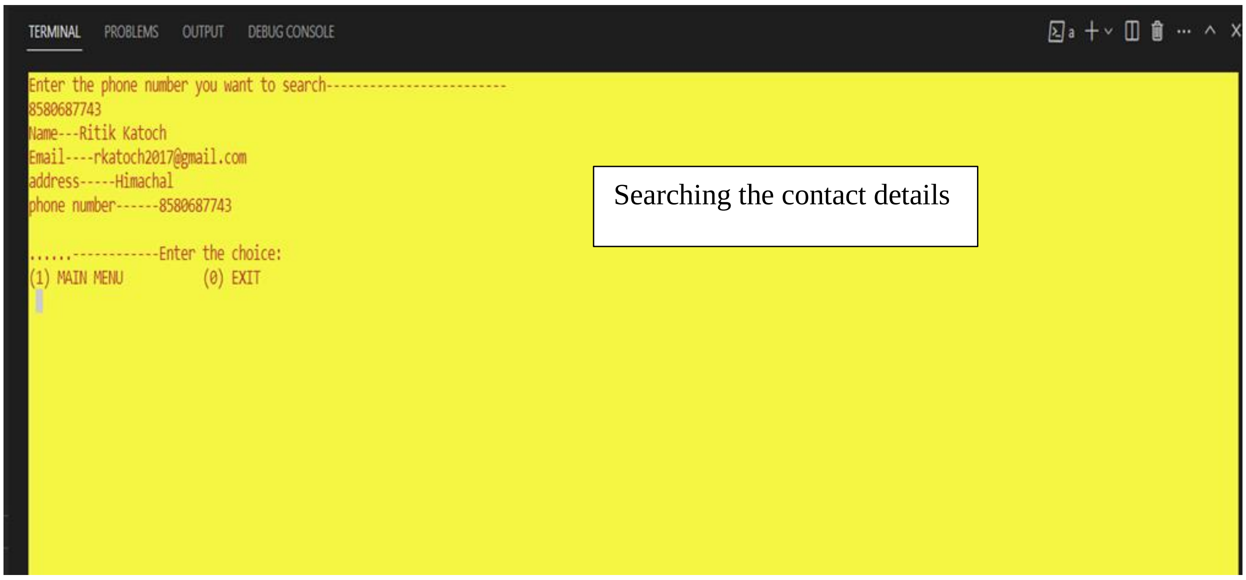
```
****************** Welcome to Ritik contacts list****************************
```



```
Contact------------------------------------ Ritik katoch

 Main menu
-----------------------------------------------------
Enter 1 for adding a new contact---------------
Enter 2 for listing all the contacts------------
Enter 3 for searching the contact---------------
Enter 4 for to delete an contact------------------
Enter 5 for to edit an contact----------------
Enter 0 for an exit-----------------------

-----------------------------------------------------
Enter your choice********************************
1
```

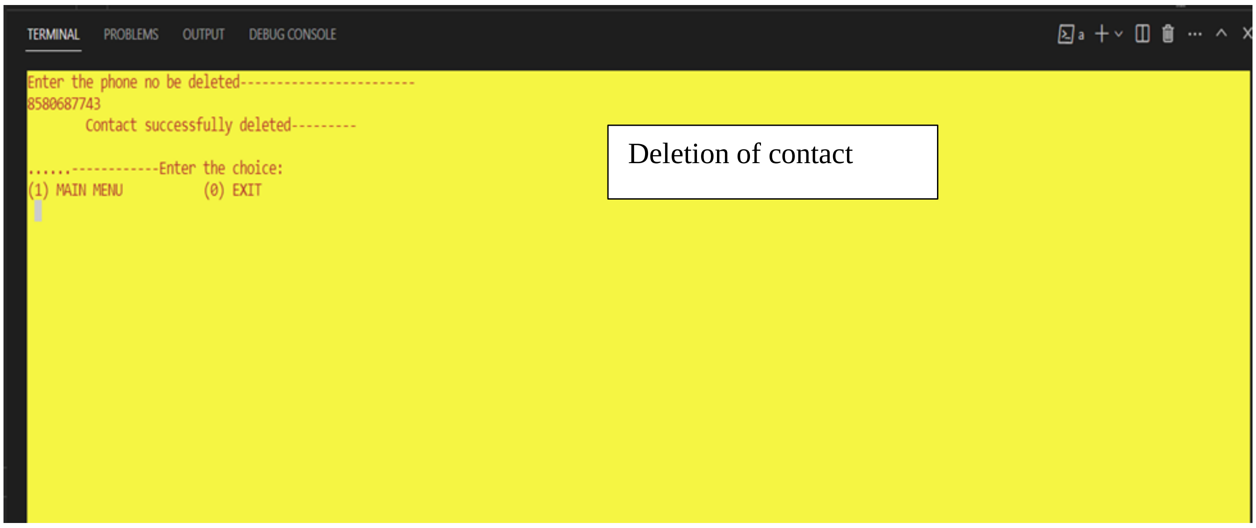


```
Enter your first name---------------
Ritik
Enter your last name-----------------
Katoch
Enter the phoneno-------
8580687743
Enter your email address---------------
rkatoch2017@gmail.com
Enter your address details-------------
Himachal
Any other contact details to add?(Y/N)------------
N
Contacts is successfully saved and created--------------


......------------Enter the choice:
(1) MAIN MENU          (0) EXIT
```



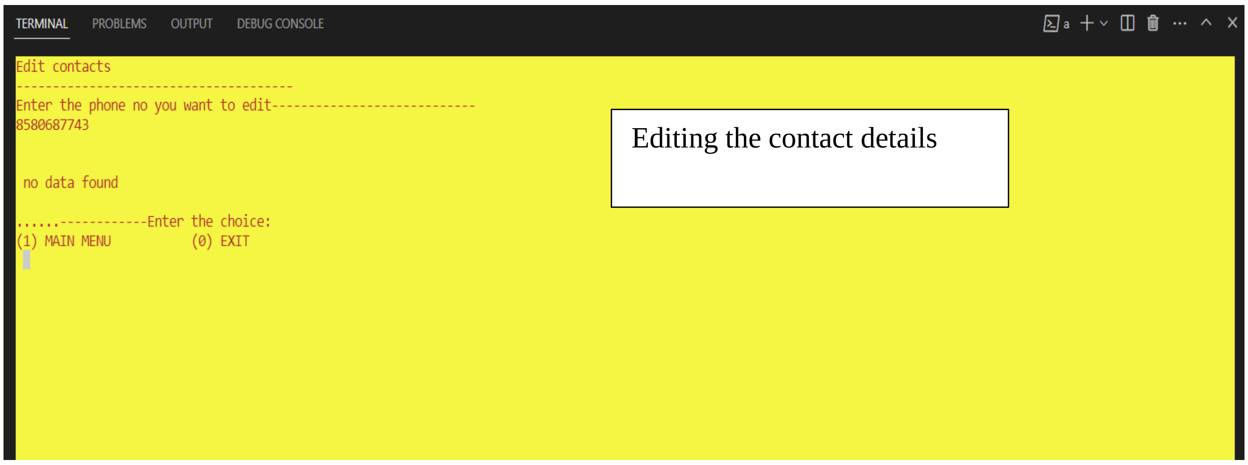
```
Enter the phone number you want to search------------------------
8580687743
Name---Ritik Katoch
Email----rkatoch2017@gmail.com
address-----Himachal
phone number------8580687743

......------------Enter the choice:
(1) MAIN MENU          (0) EXIT
```

Searching the contact details



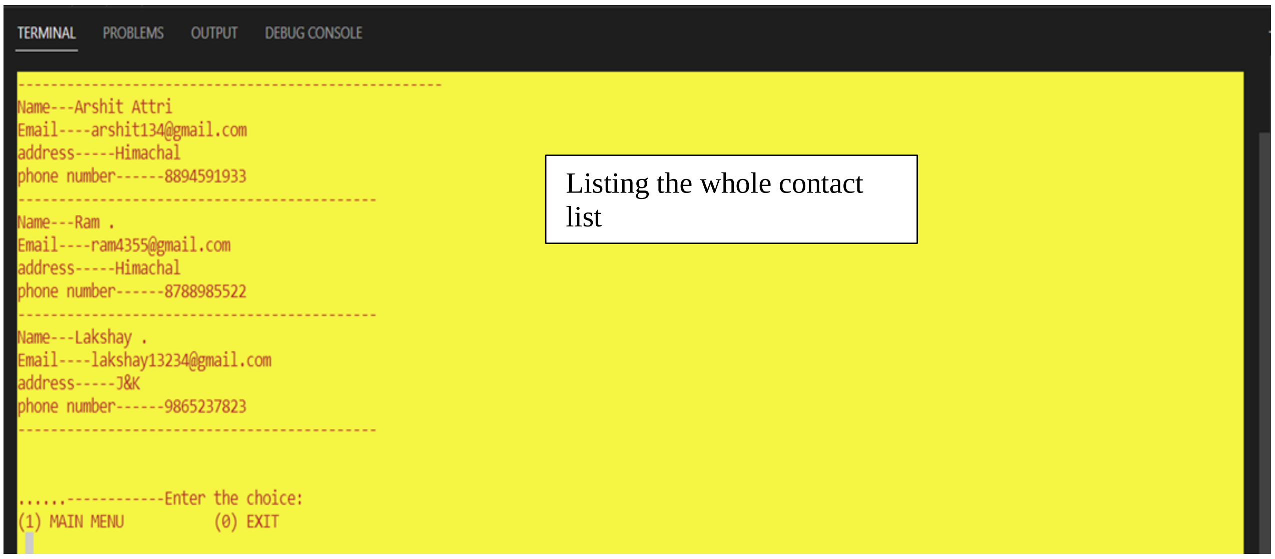
```
Enter the phone no be deleted-----------------------
8580687743
        Contact successfully deleted---------

......------------Enter the choice:
(1) MAIN MENU          (0) EXIT
```

Deletion of contact



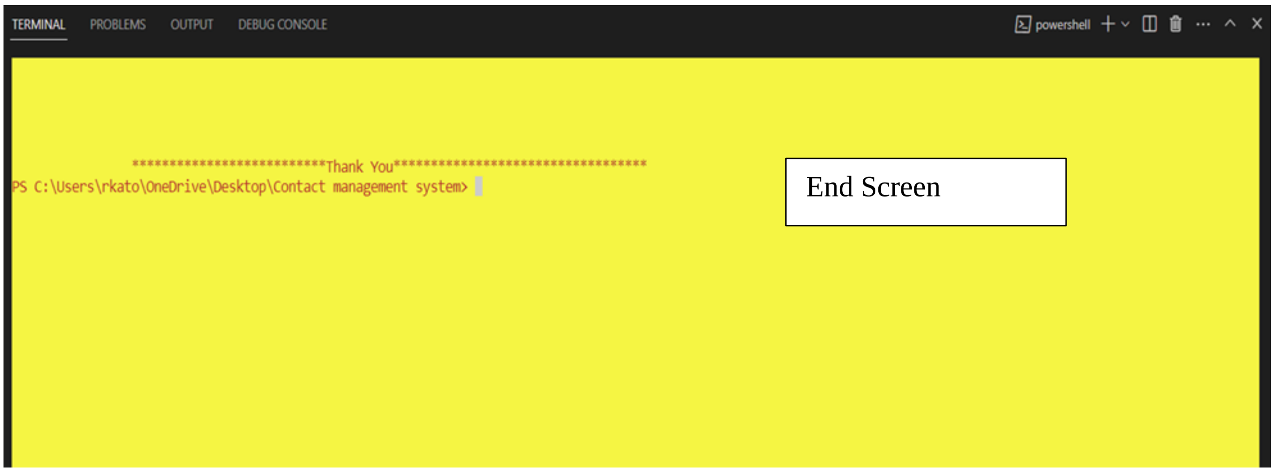
```
Edit contacts
-------------------------------------
Enter the phone no you want to edit--------------------------
8580687743


 no data found

......------------Enter the choice:
(1) MAIN MENU          (0) EXIT
```

Editing the contact details



```
----------------------------------------------------
Name---Arshit Attri
Email----arshit134@gmail.com
address-----Himachal
phone number------8894591933
--------------------------------------------
Name---Ram .
Email----ram4355@gmail.com
address-----Himachal
phone number------8788985522
--------------------------------------------
Name---Lakshay .
Email----lakshay13234@gmail.com
address-----J&K
phone number------9865237823
--------------------------------------------


......------------Enter the choice:
(1) MAIN MENU           (0) EXIT
```

Listing the whole contact list



```
               *************************Thank You********************************
PS C:\Users\rkato\OneDrive\Desktop\Contact management system>
```

End Screen

Learning Outcomes

Programming is all about data structures and algorithms. Data structures are used to hold data while algorithms are used to solve the problem using that data.

Data structures and algorithms (DSA) goes through solutions to standard problems in detail and gives you an insight into how efficient it is to use each one of them. It also teaches you the science of evaluating the efficiency of an algorithm. This enables you to choose the best of various choices.

For example, you want to search your roll number in 30000 pages of documents, for that you have choices like Linear search, Binary search, etc.

So, the more efficient way will be Binary search for searching something in a huge

number of data. So, if you know the DSA, you can solve any problem efficiently.

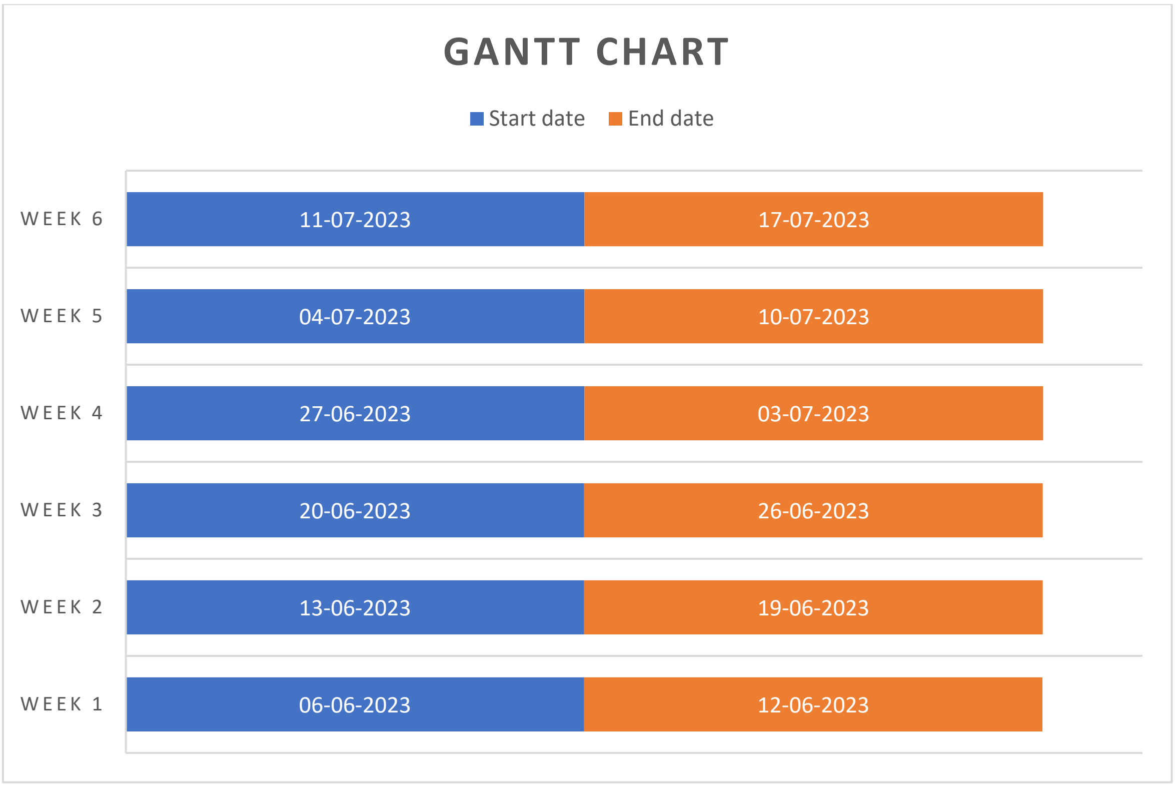
The main use of DSA is to make your code scalable because

• Time is precious

• Memory is expensive

Gantt chart

Week	Day and Date	Time	Topic
Week1	June 6, 2023 To June 12, 2023	6:30 PM-8:30 PM	Introduction of Data Structures, Complexity Analysis, Basics of Array Searching and Insertion and Solving Competitive Coding Problems on Array
Week2	June 13, 2023 To June 19, 2023	6:30 PM-8:30 PM	Sorting, Merging and Deletion Operations on Array and Solving Competitive Coding Problems on Array.
Week3	June 20, 2023 To June 26, 2023	6:30 PM-8:30 PM	Basics of Linked List, Searching, Traversing and Insertion/ Deletion in a Linked List and Solving Competitive Problems on Linked List.
Week4	June 27, 2023 To July 3, 2023	6:30 PM-8:30 PM	Fundamentals of Stack and Queue, Using Recursion and Solving Competitive Coding Problems using Stack, Queue and Recursion.
Week5	July 4, 2023 To July 10, 2023	6:30 PM-8:30 PM	Introduction of Tree and Heap, Basic Terminologies and Solving Competitive Coding Problems on Binary Trees.
Week6	July 11, 2023 To July 17, 2023	6:30 PM-8:30 PM	Introduction of Graph, Basic Terminologies and Graph Traversal using BFS and DFS Project Work.


GANTT CHART
Start date
End date
WEEK 6
11-07-2023
17-07-2023
WEEK 5
04-07-2023
10-07-2023
WEEK 4
27-06-2023
03-07-2023
WEEK 3
20-06-2023
26-06-2023
WEEK 2
13-06-2023
19-06-2023
WEEK 1
06-06-2023
12-06-2023

Project Legacy

Technical and Managerial lessons learnt:

Technical lessons:

1. **Data Import and Export:** Users often need to migrate their existing contact data into the contact management system or export it for other purposes. Learning about data import/export formats and techniques can facilitate seamless transitions.

2. **Search and Filtering:** Implementing efficient search and filtering capabilities within the contact management system is important for users to quickly locate specific contacts or information. Learning about indexing, search algorithms, and database optimization can improve performance.

Managerial lessons:

Requirement Analysis: Properly understanding the needs of the users and the business is fundamental. Learning to gather, document, and prioritize requirements helps in building a contact management system that aligns with the organization's goals.

BIBLIOGRAPHY

1. GEEKS FOR GEEKS

2.Tutorials point

3. You Tube

4. Google

5. ChatGPT

6.Power point presentations[lecture notes]