1.Implementation of different sorting techniques.

*********Bubble Sort***********

```
#include<iostream>
using namespace std;
int main ()
{
        int n;
        cout<<"Enter the size of the array";
        cin>>n;
        int a[n];
        for (int i = 0; i < n; i ++)
        {

                cin>>a[i];
        }

        for(int i=0; i<n; i++)
        {
                for (int j = 0; j<n-1-i; j++)
                {
                        int temp;
                        if( a[j] > a[j+1] )
                        {
                                temp = a[j];
                                a[j] = a[j+1];
                                a[j+1] = temp;
                        }
                }

                for(int k=0; k<n; k++)
                {

                        cout<<" "<<a[k];
                } cout<<endl;
        }

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

*********Selection Sort***********

```
#include<iostream>
```

```
using namespace std;
int main()
{
        int n;
        cout<<"Enter size of array : ";
        cin>>n;
        int arr[n];
        for(int i=0;i<n;i++)
        {
                cin>>arr[i];
        }
        for(int i=0;i<n-1;i++)
        {
                for(int j=i+1;j<n;j++)
                {
                        if(arr[j]<arr[i])
                        {
                                int temp = arr[j];
                                arr[j] = arr[i];
                                arr[i] = temp;
                        }
                }
                for(int k=0;k<n;k++)
                {
                        cout<<" "<<arr[k];
                }
                cout<<endl;
        }
        cout<<"Sorted array : ";
        for(int i=0;i<n;i++)
        {
                cout<<" "<<arr[i];
        }
}
```

*********Insertion Sort***********

```
#include<iostream>
using namespace std;
int main()
{
        int n;
        cout<<"enter the size for array: ";
        cin>>n;
        int a[n];
        for (int i=0;i<n;i++)
        cin>>a[i];
        for(int i=0;i<n-1;i++)
```

```
        {
                for(int j=i+1;j>0;j--)
                {
                        int temp;
                if(a[j]<a[j-1])
                {
                            temp=a[j];
                                a[j]=a[j-1];
                                a[j-1]=temp;
                        }
                }
                for (int k=0;k<n;k++)
                        cout<<" "<<a[k];
        cout<<endl;
    }
        cout<<"sorted array"<<endl;
        for(int i=0;i<n;i++)
            cout<<" "<<a[i];
        return 0;
}


*********Radix Sort***********


#include<iostream>
using namespace std;

//Function to get the largest element from an array
int getMax(int array[], int n)
{
  int max = array[0];
  for (int i = 1; i < n; i++) if (array[i] > max)
      max = array[i];
  return max;
}

//Using counting sort to sort the elements in the basis of significant places
void countSort(int array[], int size, int place)
{
  const int max = 10;
  int output[size];
  int count[max];

  for (int i = 0; i < max; ++i)
    count[i] = 0;

  //Calculate count of elements
  for (int i = 0; i < size; i++)
    count[(array[i] / place) % 10]++;
```

```
  //Calculating cumulative count
  for (int i = 1; i < max; i++)
    count[i] += count[i - 1];

  //Placing the elements in sorted order
  for (int i = size - 1; i >= 0; i--)
  {
    output[count[(array[i] / place) % 10] - 1] = array[i];
    count[(array[i] / place) % 10]--;
  }

  for (int i = 0; i < size; i++)
    array[i] = output[i];
}

//Main function to implement radix sort
void radixsort(int array[], int size)
{
  //Getting maximum element
  int max = getMax(array, size);

  //Applying counting sort to sort elements based on place value.
  for (int place = 1; max / place > 0; place *= 10)
    countSort(array, size, place);
}

//Printing an array
void display(int array[], int size)
{
  int i;
  for (i = 0; i < size; i++)
    cout << array[i] << "\t";
  cout << endl;
}

int main()
{
  int array[] = {112, 400, 543, 441, 678, 675, 9, 777};
  int n = sizeof(array) / sizeof(array[0]);
  cout<<"Before sorting \n";
  display(array, n);
  radixsort(array, n);
  cout<<"After sorting \n";
  display(array, n);
}


*********Quick Sort***********
```

```
#include <iostream>

using namespace std;

void quick_sort(int *array, int low, int high) {
    if (low < high) {
        int pivot = array[high];
        int i = low - 1; // This finds smallest element of array

        for (int j = low; j < high; j++)
                {
            if (array[j] <= pivot) {
                i++;
                swap(array[i], array[j]);
            }
        }

        swap(array[i + 1], array[high]);

        quick_sort(array, low, i);
        quick_sort(array, i + 2, high);
    }
}

int main() {
    int array[] = {5, 3, 2, 1, 4};
    int n = sizeof(array) / sizeof(array[0]);

    quick_sort(array, 0, n - 1);

    for (int i = 0; i < n; i++) {
        cout << array[i] << " ";
    }

    cout << endl;

    return 0;
}


*********Shell Sort***********

#include <iostream>
using namespace std;

void shellSort(int arr[], int size)
{
    // Start with a big gap, then reduce the gap
    for (int gap = size / 2; gap > 0; gap /= 2)
    {
```

```
        // Perform insertion sort on the subarrays defined by the gap
        for (int i = gap; i < size; i++)
        {
            int temp = arr[i];
            int j;
            for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)
            {
                arr[j] = arr[j - gap];
            }
            arr[j] = temp;
        }
    }
}

int main() {
    int n;
        cout<<"enter the size for array: ";
        cin>>n;
        int arr[n];
        for (int i=0;i<n;i++)
         cin>>arr[i];
    int size = sizeof(arr) / sizeof(arr[0]);

    cout << "Before sorting: ";
    for (int i = 0; i < size; i++)
    {
        cout << arr[i] << " ";
    }
    cout << endl;

    shellSort(arr, size);

    cout << "After sorting: ";
    for (int i = 0; i < size; i++)
    {
        cout << arr[i] << " ";
    }
    cout << endl;
}
```

2. Implementation of different searching techniques.

*********To implement linear search**********

```
#include<iostream>
using namespace std;
 int main()
 {
```

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

        int a[n];
        cout<<"Enter "<<n<<" elements : ";
        for(int i=0; i<n; i++)
        {
                cin>>a[i];
        }
        cout<<"You have entered the array"<<endl;

        for(int i=0; i<n; i++)
        {
                cout<<" "<<a[i];
        }
        cout<<endl;
        cout<<"Enter the element you want to search : ";

        int s;
        cin>>s;
        for(int i=0; i<n; i++)
        {
                if(a[i]==s)
                {
                        cout<<"Element found at index "<<i<<endl;
                        break;
                }

                if(i==n)
                {
                        cout<<"Element not found in array"<<endl;
                }
        }
        return 0;
}


*********To implement binary search**********



#include <iostream>
using namespace std;

int binarySearch(int arr[], int l, int r, int x) {
    while (l <= r) {
        int mid = l + r / 2;
        if (arr[mid] == x) {
            return mid;
```

```
        } else if (arr[mid] < x) {
            l = mid + 1;
        } else {
            r = mid - 1;
        }
    }
    return -1;
}

int main() {
    int n;
    cout << "Enter the number of elements in the array: ";
    cin >> n;

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

    int x;
    cout << "Enter the element to search for: ";
    cin>>x;


    int result = binarySearch(arr, 0, n - 1, x);
    if (result == -1)
        {
        cout << "Element not found" << endl;
    }
        else
        cout << "Element found at index " << result << endl;

    return 0;
}
```

3. Implementation of Stacks (Using arrays and Linked List)

*********To implement Stack using array**********

```
#include<iostream>
using namespace std;
#define MSize 25
int stack[MSize];
int top =-1;

void push()
```

```
{
        int item;
        if(top==MSize-1)
        {
                cout<<"Stack Full\n";
        }
        else
        {
                cout<<"Enter values to be inserted : ";
                cin>>item;
                stack[++top]=item;
        }
}
void pop()
{
        int item;
        if(top==-1)
        {
                cout<<"Empty Stack"<<endl;
        }
        else
        {
                item=stack[top];
                top=top-1;
                cout<<"Deleted Element : "<<endl;
        }
}

void traverse()
{
        if(top==-1)
        {
                cout<<"Empty Stack"<<endl;
        }
        else
        {
                cout<<"Values in stack : "<<endl;
        }

        for(int i=top; i>0;i--)
        {
                cout<<"\t"<<stack[i];
        }
}
int main()
{
        int choice;
        char ch;
        do
        {
```

```
                cout<<"******Stack Operation*******\n\n";
                cout<<"1-Push value\n\n2- Pop Value\n\n3- Traverse\n\n4- Exit\n";
                cin>>choice;
                switch(choice)
                {
                        case 1:
                                push();
                                break;
                        case 2:
                                pop();
                                break;
                        case 3:
                                traverse();
                                break;
                        default:
                                cout<<"\n Invalid Choice";
                }
                cout<<"\n Enter (Y/y) to continue : ";
                cin>>ch;
        }
        while(ch=='Y'||ch=='y');
        return 0;
}
```

*********To implement Stack using linked list***********

```
#include <iostream>
#include <stack>

using namespace std;

struct Node {
    int data;
    Node* next;

    Node(int val) : data(val), next(nullptr) {}
};

class LinkedList {

private:
        Node* head;

public:
    LinkedList() : head(nullptr) {}

    void push(int val) {
        Node* newNode = new Node(val);
```

```
        if (head == nullptr)
                {
            head = newNode;
        }
        else {
            newNode->next = head;
            head = newNode;
        }
    }

    void pop() {
        if (head == nullptr) {
            cout << "Stack is empty. Cannot pop." << endl;
            return;
        }
        Node* temp = head;
        head = head->next;
        delete temp;
    }

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

int main() {
    LinkedList stackLinkedList;

    int choice, value;
    char continueInput;

    do {
        cout << "1. Push" << endl;
        cout << "2. Pop" << endl;
        cout << "3. Print" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter value to push: ";
                cin >> value;
                stackLinkedList.push(value);
                break;
            case 2:
```

```
                stackLinkedList.pop();
                break;
            case 3:
                cout << "Stack: ";
                stackLinkedList.print();
                break;
            default:
                cout << "Invalid choice." << endl;
        }

        cout << "Do you want to continue (y/n)? ";
        cin >> continueInput;

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

    return 0;
}
```