NAME : Yuvraj Singh

SUBJECT : DSA

CLASS : BCA 2nd SEM

FACULTY : Dinesh Prajapat

YEAR : 2024-25

INDEX

S NO	TITLE	DATE OF EXP	DATE OF SUB	SIGN
1	WAP to implement bubble sort with algorithm	26-04-25	06-05-25	
2	WAP to implement selection sort with algorithm	26-04-25	06-05-25	
3	WAP to implement insertion sort with algorithm	26-04-25	06-05-25	
4	WAP to implement merge sort with algorithm	26-04-25	06-05-25	
5	WAP to implement quick sort with algorithm	26-04-25	06-05-25	
6	Design and implement stack & its operation using array	26-04-25	06-05-25	
7	Design & implement queue & its operation using array	26-04-25	06-05-25	

NAME : Yuvraj Singh

SUBJECT : DSA

CLASS : BCA 2nd SEM

FACULTY : Dinesh Prajapat

YEAR : 2024-25

INDEX

S NO	TITLE	DATE OF EXP	DATE OF SUB	SIGN
Q.8	Design and implement linked list and its O/P using arrays		14-06-25	
Q.9	Write a program to create tree using linked list		14-06-25	
Q.10	Write a program to represent a graph using adjacency matrix		14-06-25	
Q.11	Write a program to implement linear search		14-06-25	
Q.12	Write a program to implement binary search.		14-06-25	

Neel Kamal
PAGE NO.: /
DATE / /

Q.1 Write a program to implement bubble sort with algorithm.

Start

Input array size

- input the size of array from user
- store the size in the variable size(n)

Input array elements :

- input the elements from user (array)
- store in arr.

Bubble sort function :

- Display the unsorted array

In bubbleSort →

- Iterate through the array from 0 to (n-2) times
  - for each index j from 0 to (n-i-2)
    - if arr[j] > arr[j+1]
      swaping them

- After each pass, print the array
- for each index k from 0 to (n-1)
  - print arr[k] followed by a space

- print the sorted array for each index i from 0 to n-1.

- we can initialise a boolean flag swapped to false if any swapping occur this will be modified to true & if no swapping occurs we can early terminate ~~Teacher's Sign / Sorting~~

END

OUTPUT ⇒

Original array = 64 34 25 12 22 11 90

sorted array = 11 12 22 25 34 64 90

```
#include <iostream>
using namespace std;
void bubblesort (int arr[], int n)
{
    for(int i=0; i<n-1 ; i++)
    {
       bool swapped = false;
    for(int j=0; j<n-i-1; j++)
    {
      if (arr[j] > arr[j+1])
      {
          swap(arr[j] ; arr[j+1]);
          swapped = true;
    }}
          if(!swapped)
          break;
    } }

void printarray (int arr[]; int n)
    {
        for (int i=0; i<n ; i++)
        cout << arr[i] << " ";
         cout << endl;
  }  int main() {  int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n= sizeof(arr) / sizeof(arr[0]);
     cout << "original array";
     print array(arr, n);
     bubblesort (arr, n);
```

Teacher's Signature _____

Original array =	64	25	12	22	11
After pass 1 =	11	64	25	12	22
After pass 2 =	11	12	64	25	22
After pass 3 =	11	12	22	64	25
After pass 4 =	11	12	22	25	64
Sorted array =	11	12	22	25	64

```
cout<< "sorted array = ";
printarray (arr, n);
return 0;
}
```

Q.2 Write a program to implement selection sort with algorithm.

Algorithm : 1) start

2) Repeat for i=0 to n-2;
- Assume the current index i is the minimum
  minidex = i
- Repeat for j= i+1 to n-1;
- IF A[j] < A[minidex], then set minidex = j
- swap A[i] with A[minidex]

3) End

Code :-

```
#include <iostream>
using namespace std;
void selectionsort (int arr[], int n)
{
    for (int i=0; i<n-1; i++)
    {
        for(int j= i+1; j<n; j++)
        {
```

```
if (arr[i] < arr[minidex])
{
}
}
if(minidex != i) {
    int temp = arr[i];
arr[i] = arr[minidex];
 arr[minidex] = temp;
}
     cout << "After pass" << i+1 << ":";
    for (int k=0; k<n; k++)
    {
       cout << arr[k] << " "; }
     cout << endl;
}
}
 void printarray (int arr[], int n)
  {
     for (int i=0; i<n; i++)
          cout << arr[i] << " ";
          cout << endl;
   }
int main() {
     int arr[] = {64, 25, 12, 22, 11};
      int n= sizeof(arr) / sizeof(arr[0]);
      cout << " original array = ";
```

```
printarray (arr, n);
selectionsort (arr, n);
 cout << "sorted array = ";
  printarray (arr, n);
return 0;
}
```

Q.3 Write a program to implement insertion sort with algorithm.

Algorithm :- 1) start with second element (index 1) assuming first already sorted

2) • for (i = 1 to n-1)
compare current key
• key = A[i]
j = i-1

3) shift all the elements
while j >= 0 & A[j] > key :
A[j+1] = A[j]

4) insert key at correct position

5) Repeat steps 2-4 for each element
A[j+1] = key ;

Original array = 12 11 13 5 6

Sorted array = 5 6 11 12 13

Code :

```
#include<iostream>
using namespace std;
void insertionsort(int arr[], int n)
{
    for(int i=1; i<n; i++)
    {
      int key = arr[i];
      int j= i-1;
while(j>=0 && arr[j] > key) {
   arr[j+1] = arr[j];
    j--;   }
arr[j+1]= key;
}  }
    void printarray(int arr[], int n) {
    for(int i=0; i<n; i++)
       cout<<arr[i]<<" ";
       cout<<endl;   }
int main() {   int arr[] = {12,11,13,5,6};
            int n= sizeof(arr) / sizeof(arr[0]);
       cout<<" original array =";
       printarray(arr, n);
      insertionsort(arr, n);
   cout<<"sorted array =";
    printarray(arr,n);
   return 0;
}
```

Teacher's Signature _____

Q.4 Write a program to implement merge sort with algorithm.

Algorithm: 1) Divide the array into two halves
2) Recursively sort both halves
3) Merge the two sorted halves into a single sorted array.

Code :-

```
#include<iostream>
using namespace std;
void merge(int arr[], int left, int mid, int right)
{
    int n1 = mid - left + 1;
    int n2 = right - mid;
    int L[n1], R[n2];

    for(int i=0; i<n1; i++)
        L[i] = arr[left + i];
    for(int j=0; j<n2; j++)
        R[i] = arr[mid + 1+j];

    int i=0; j=0, k=left;
    while(i<n1 && j<n2)
    {
        if(L(i] <= R[j])
            arr[k++] = L[i++];
    else
```

Teacher's Signature _____

Original array = 12 11 13 5 6 7

sorted array = 5 6 7 11 12 13

```
arr[k++] = R[j++];
}
  while (i<n1)
      arr[k++] = L[i++];
  while (j<n2)
          arr[k++] = R[j++];
}
void mergesort (int arr[], int left, int right)
{
      if (left < right)
      {   int mid = left + (right - left)/2;
      mergesort (arr, left, mid);
      mergesort (arr, mid+1, right);
      merge (arr, left, mid, right);
}}
   void printarray (int arr[], int size) {
         for (int i=0; i<size; i++)
          cout << arr[i] << " " << endl;
   }
int main() {   int arr[] = {12, 11, 13, 5, 6, 7};
          int size = sizeof(arr)/sizeof(arr[0]);
          cout << " original sort = ";
          printarray(arr, size);
          mergesort (arr, 0, size-1);
          cout << " sorted array = ";
          printarray (arr, size);
          return 0;
   }
```

Teacher's Signature _____

Q.5 Write a program to implement quick sort with algorithm.

Algorithm :- 1) choose a Pivot : Usually the last element in current segment
· if low < high then

2) Partition : Rearrange of array
Pivot index ← Partition (arr, low, high)
Quicksort (arr, low, pivotindex -1)

3) Recursion : Quicksort (arr, pivotindex +1, high)

Code :-

```
#include <iostream>
using namespace std;
void swap (int &a, int&b) {
    int temp=a;
     a=b;
     b=temp;  }
int partition(int arr[], int low, int high)
   {    int pivot = arr[high];
        int i = low-1;
  for (int j= low; j<= high-1; j++) {
  if (arr[j] <= pivot) {
                        i++;
                        swap(arr[i], arr[j]);
                      }
  }
```

Teacher's Signature _____

Original array = 10 7 8 9 1 5

sorted array = 1 5 7 8 9 10

```
swap(arr[i+1], arr[high]);
    return (i+1);
}
void quicksort (int arr[], int low, int high)
  {
      if (low < high) {
         int pi = partition (arr, high, low);
         quicksort (arr, low, pi-1);
          quicksort (arr, pi+1, high);
} }
    void printarray (int arr[], int size)
    {
        for (int i=0; i<size ; i++)
          cout<< arr[i] <<" " << endl;
}
 int main() {
          int arr[] = { 10,7,8,9,1,5 };
          int n = sizeof (arr) / sizeof (arr[0]);
          cout<<" original array = ";
          printarray (arr, n);
          quicksort (arr, 0, n-1);
           cout << " Sorted array = ";
           printarray (arr, n);
    return 0;
}
```

Teacher's Signature _____

Q.6 Design and implement stack and its operation using arrays.

Algorithm :

1) Initialize stack = top ← -1

2) Push operation - if top == max - 1
then output = "stack overflow"
else = top ← top + 1
stack [top] ← value
output value + "push to stack"

3) Pop operation = if top == -1 then
output = "stack underflow"
else = output
stack [top] + "popped from stack"

4) Peek operation = if top == -1 then
output = "empty"
else output = "Top element is" + stack [Top]

5) isempty operation = if top == -1 then
return true
else return false

Teacher's Signature _____

```
10 Pushed to stack
20 Pushed to stack
30 Pushed to stack
stack element = 30 20 10
Top element is = 30
 30 popped from stack
 stack element = 20 10
```

Code :

```
#include<iostream>
using namespace std;
# define MAX 100
 class stack {
  Private:
          int arr[MAX];
          int top;
 public:
      stack() {
      top = -1;
   }
  bool isempty() {
          return top == MAX-1;
           }
 void push (int value) {
       if (isfull())
       {     cout<<" stack overflow "<< endl;
       } else {
             arr[++top]= value;
    cout << value << " pushed to stack " << endl; }
}
  void pop() {
          if(isempty()) {
              cout<<" stack undurflow"<<endl;
         } else {
    cout << arr[top--] <<" popped from stack" << endl;
} }
```

Teacher's Signature _____

```
int peep() {
        if (isempty()) {
            cout << " stack is empty " << endl;
        return -1;
    } else {
            return arr[top];
    } }
    void display() {
     if (isempty()) {
            cout << " stack is empty " << endl;
    } else {
                Cout << " stack elements";
    for(int i= top; i>=0; i--)
    {     cout << arr[i] << " ";
    }  cout << endl;
    }
}
};
 int main()
{    stack s;
     s.push(10);
     s.push(20);
     s.push(30);
     s.display();
    cout << " Top element is " << s.peep() << endl;
    s.pop();
    s.display();
    return 0;
}
```

Teacher's Signature _____

Q.7 Design and implement Queue and its operations arrays.

Algorithm :

1) enqueue :- • if size == capacity
• rear = (rear + 1) % capacity
• set array [rear] = item

2) Dequeue :- size == 0
item = array [front]
• front = (front + 1) % capacity
• size--.

3) Peek :- • Size == 0
• Return array [front]

4) Display :-
• size == 0
• set index = front
• for i = 0 to i < size;
• print array [index];
• Index = (index + 1) % capacity

Code :-

```
#include <iostream>
using namespace std;
    class Queue {
    private:
            int front, rear, size;
            int capacity;
```

Teacher's Signature _____

En queued = 10
Enqueued = 20
enqueued = 30

Queue contents = 10 20 30

Dequeue contents = 20 30

Queue contents = 20 30

front item is = 20

```
    int *array;
public:
        Queue(int cap)
        {  capacity = cap;
           front = 0;
           rear = -1;
           size = 0;
           array = new int[capacity]; }
    Queue()
    {    delete[] array; }
    bool is empty () {
                return size == 0; }
    bool isfull() {
                return size == capacity;
              }
void enqueue (int item)
{           if (isfull())
        {       cout<<" Queue is full. can't enqueue \n";
        return; }
    rear = (rear+1) % capacity;
     array[rear] = item;
     size++;
     cout << " Enqueued" << item << endl;
}
```

Teacher's Signature _____

```
int dequeue() {
        if( isempty())
        {
          cout << " Queue is empty, can't dequeue \n";
          return -1;
        }
  int item = array [front];
  front = (front + 1) % capacity;
  size--;
   cout << "Dequeued=" << item << endl;
   return item;
  }
int peek() {
        if (isempty())
        {
          cout << " Queue is empty. \n";
          return;
        }
     cout << " Queue contents = ";
     int index = front;
  for (int i=0; i<size; i++)
  {
     cout << array[index] << " ";
      index = (index+1) % capacity;   }
      cout << endl;  }
};
```

```
int main()
{
    Queue q(5);
    q.enqueue(10);
    q.enqueue(20);
    q.enqueue(30);

    q.display();
    q.dequeue();
    q.display();

cout << " front item is = " << q.peek() << endl;
return 0;
}
```

OPERATION OF LINK LIST

1. Add to list
2. Delete from list
3. Print List
4. Quit

Enter your choice :- 1

Enter a value (quite to stop) :- 32

Enter a value (quite to stop) :- 56

Enter a value (Quite to stop) :- 87

Enter a value (Quite to stop) :- quite

1. Add to list
2. Delete from list
3. Print list
4. Quite

Enter your choice :- 2

Which value do you want to delete from the list ? 32

1. Add to list
2. Delete from list
3. Print list
4. Quite

Enter your choice :- 3

Entered list elements are :- 56 87

Neel Kamal
PAGE NO.: 1
DATE / /

Q.8 C++ program to design and implement linked list and operations using array :

Algorithms :

I Initialization

Define a Node structure with
- data → stores string value
- next → stores index of next node

② Creat an array list[] of Node to store the linked list

③ Initialize :
- head = -1 (represents an empty list)
- freeIndex = 0 (tracks next free slot in the array)

④ set list[i].next = -1 for all i (marks end of links initially)

II Add to list operation

1. Repeat untill user enters "quit" :
   - Ask for input value
   - If input is "quit", stop
   - If freeIndex >= size of array → list is full → display an error
   - store value at list[freeIndex].data

Teacher's Signature _____

- set list[freeIndex].next = head.
- Update head = freeIndex
- Increment freeIndex.

III Delete from List operation

1. Ask the user which value to delete
2. If head == -1, the list is empty → display message
3. Set current = head and prev = -1.
4. Loop through the list:
   - If list[current].data == value
     - If prev == -1, update head = list[current].next (deleting head)
     - Else, link previous node:
       list[prev].next = list[current].next
     - Set list[current].next = -1 to mark node as deleted
     - stop deletion.
5. If value not found, display message.

IV Print list operation

1. If head == -1, display "Nothing is in the list"
2. Start from current = head.
3. while current != -1:
   - Print list[current].data
   - Move to current = list[current].next.

Teacher's Signature _____

Code

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

struct Node
{
    string data;
    int next;
};

class LinkedList
{
    private:
    static const int MAX_SIZE = 100;
    Node list[MAX_SIZE];
     int head;
     int freeIndex;
Public:
    LinkedList()
    {    head = -1;
         freeIndex = 0;
      for (int i=0; i< MAX_SIZE ; i++)
        {
            list[i].next = -1
        }
}
```

Teacher's Signature _____

```
void addToList()
{
    string temp;
    bool quite = false;
    while (!quite)
    {
        quite = true;
    } else {
        if(FreeIndex >= MAX_SIZE)
        {
        cout << " List is full !" << endl;
        return;
        }
    list[freeIndex].data = temp;
    list[freeIndex].next = head;
    head = freeIndex;
    freeIndex++;
    }
}
}
void deleteFromList()
{
    if(head == -1)
    {
        cout << " Nothing is in the List " << endl;
        return;
    }
```

```
string deletion;
cout<<" which value do you want from the list ? ";
cin>> deletion ;

int prev = -1;
int current = head;

  while ( current != -1)
{
    if( list[current].data == deletion)
     {
         if( prev == -1)
         {
            head = list[current].next ;
         }else{
                list[prev].next = list[current].next ;
              }
       list[current].next = -1;
        return ;
    }
   cout<<" That value is not in the list" << endl;
}
  void printList ()
   {
       IF( head == -1)
       {
          cout<<" Nothing is in the list" << endl;
          return ;
       }
```

```
cout << " Entered list elements are :" << endl;
  int current = head;
  while ( current != -1)
  {
     cout << list[current].data << " ";
     current = list[current].next ;
  }

   cout << endl;
  }
};  int main() {
      Linkedlist list ;
      bool quite = false;
      int choice ;
     cout << " OPERATION OF LINK LIST " << endl;
while(!quite)
{
        cout << " \n 1. Add to List " << endl;
        cout << " 2. Delete from list" << endl;
        cout << " 3. Print List" << endl;
        cout << " 4. Quit" << endl;
        Cout << " Enter your choice :- ";
        cin >> choice ;

switch(choice)
  {
        case 1 :
            list.addToList();
            break;
```

```
case 2:
list.deleteFromList();
break;

case 3:
list.printList();
break;          case 4:
                    quite = true; break;
default:

cout << " That is not a valid input, quitting
      program" << endl;

quite = true;
}
}
    return 0;

}
```

TREE OPERATIONS USING LINKED LIST

1. Add a child
2. Delete a node
3. Print tree (Preorder)
4. Quit

Enter your choice :- 1

Enter parent value (or 'none' if adding root) : A

Enter child value :- B

Added B as a child of A.

1. Add a child
2. Delete a node
3. Print tree (Preorder)
4. Quit

Enter your choice :- 3

Preorder traversal of the tree :- AB

1. Add a child
2. Delete a node
3. Print tree (preorder)
4. Quit

Enter your choice :- 4

Quitting program.

Q.9 Write a program to create Tree using Linked List.

Algorithm

1. Initialization
   - Define a childNode structure with:
     child → pointer to a tree node
     next → pointer to the next child in linked list.
   - Define a TreeNode structure with:
     data → string value
     children → pointer to the linked list of children
   - Initialize the tree
     set root to nullptr (empty tree)

2. Insert operation (Add a child)
   - Input: parentVal (parent value), childVal (child value)
   - Create a new TreeNode with data = childVal.
   - If root is nullptr:
     - set root to the new node
     - Display "Root created with value: childVal"
     - stop.
   - find the parent node with value parentVal using preorder traversal
     - If not found, display "Parent not found" and delete the new node
     - stop

Teacher's Signature _____

- Create a new childNode pointing to the TreeNode
- Add the childNode to the front of the parent's children linked list.
- Display "Added childVal as a child of parentVal"

3. Delete operation (Delete a Node)
   - Input - Value (value to deleted)
   - If root is nullptr, display "Tree is empty!" and stop
   - If root.data == value :
     - Delete the entire subtree rooted at root
     - set root to nullptr
     - Display "Root with value deleted"
     - stop
   - Search for the parent of the node with value using level-order traversal :
     - If not found, display "Node not found" and stop
     - Unlink the node from parent's children list
     - Delete the subtree rooted at the node.
   - Display "Node with value and its subtree deleted".

4. Print Operation (Preorder Traversal)
   - If root is nullptr, display "Tree is empty"
   - Display "preorder traversal of the tree

Teacher's Signature _____

• Traversal of the tree in preorder :
  - start at node = root
  - while node is not nullptr
    • Print node.data
    • for each child in node.children, recursively traverse the child's subtree.
  • Print a newline.

5. Menu system
  • Display "TREE OPERATION USING LINKED LIST" :
  • Loop untill user quits
    • Display menu :
      - "1. Add a child"
        "2. Delete a node"
        "3. Print tree"
        "4. Quit"
        "Enter your choice :-"
    • Read users choice
- if choice is 1 call insert operation (read parent & child value)
- if choice is 2
    • Read value to delete
    • Call delete operation
- If choice is 3
    call print operation
- If choice is 4, Quit the loop

STOP

Teacher's Signature _____

Neel Kamal
PAGE NO.:
DATE / /

Code

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

struct childNode
{       struct TreeNode* child;
        childNode* next;
      childNode(TreeNode* c): child(c), next(nullptr)
                    {  }
};
Struct TreeNode
 {      string data;
        childNode* children;
        TreeNode(string val) : data(val), children(nullptr)
                                        {}
};
class Tree {
    private :
        TreeNode* root;
      TreeNode* findNode(TreeNode* node, string value)
      {
        if(node== nullptr) return nullptr;
        if(node->data == value) return node;

    childNode* current = node->children;
```

Teacher's Signature _____

```
while( current != nullptr)
{
    TreeNode * result = findNode( current->child, value);
    if( result != nullptr) return result;
    current = current -> next;
  }
  return nullptr ;
}
 void deleteSubtree (TreeNode* node)
    if (node == nullptr) return ;

    childNode* current = node->children;
    while (current != nullptr)
{
   childNode* temp= current;
    current = current -> next;
    deletesubtree (temp-> child);
     delete temp ;
  }
   delete node ;
}
void preorder( TreeNode* node)
 {
     if(node == nullptr) return ;
   cout<< node->data <<" ";
   childNode* current = node -> children;
```

```
while(current != nullptr)
 {
    preorder(current -> child);
    current = current -> next;
  }
}
public:
     Tree() : root(nullptr) {}
    void insert(string parentVal, string childVal);
   if(root == nullptr)
   {
     root = childNode;
     cout << " Root Created with value : " << childVal << endl;
     return;
}
 TreeNode* parent = findNode(root, parentVal);
   if(parent == nullptr)
   {
     cout << " Parent with value " << parentVal << " not
        found " << endl;
      delete childNode;
      return;
}
```

```
ChildNode* newChild = new ChildNode (childNode);
newChild -> next = parent -> children;
parent -> children = newChild;
cout << " Added " << childVal << " as a child of "
        << parentVal << endl;
}

void deleteNode (string value)
{
    if(root == nullptr)
    {
        cout << " Tree is empty ! " << endl;
        return;
    }
if(root -> data == value)
{
    deleteSubtree(root);
    root = nullptr;
  cout << " Root with value " << value << " deleted ! ";
    return;
}

TreeNode* parent = nullptr;
ChildNode* currentChild = nullptr;
  bool found = false;
```

```
TreeNode* queue[1000];
int front=0, rear=0;
queue[rear++] = root;

while(front < rear && !found)
{
    TreeNode* node = queue[front++];
    childNode* current = node->children;
    ChildNode* prevChild = nullptr;

    while(current != nullptr)
    {
        if(current->child->data == value)
        {
            parent = node;
            currentChild = current;
        if(prevChild == nullptr)
        {
            parent->children = current->next;
        }
        else {
            prevChild->next = current->next;
        }
        found = true;
        break;
}
```

```
prevChild = current;
  queue[rear++] = current -> child;
   current = current -> next;
     }
}
 if(!found)
{
   cout<<"Node with value "<< value <<" not
     found!" << endl;
      return;
}
 deleteSubtree(currentChild -> child);
  delete currentChild;
 cout<<"Node with value "<< value <<" And
     its subtree deleted" << endl;
}

// Perform preorder traversal to print the tree

 void printPreodr()
  {
     if(root == nullptr)
    {
       cout<<"Tree is empty!" << endl;
       return;
  }
```

```
cout << " preorder traversal of the tree" ;
preorder(root);
cout << endl;
}

~Tree() {
        deleteSubtree(root);
    }
};
int main()
{
    Tree tree;
    bool quite = false;
    int choice;
    string parentVal, childVal, deleteVal;

cout << " TREE OPERATIONS USING LINKED LIST" << endl;

while(!quite){

    cout << "\n 1. Add a child" << endl;
    cout << " 2. Delete a node" << endl;
    cout << " 3. Print tree(preorder)" << endl;
    cout << " 4. Quit << endl;
    cout << " Enter your choice : " ;
    cin >> choice;
```

```
switch (choice)
{       case 1 :
        cout << " Enter parent value (or none if adding root ";
        cin >> parentVal;
        cout << " Enter child value :";
        cin >> childVal;
        tree.insert (parentVal == "none" ? "" : parentVal, childVal);
         break;

        case 2 :
        cout << " Enter value to delete ";
        cin >> deleteVal;
        tree.deleteNode (deleteVal);
        break;

        case 3 :
            tree.printPreorder ();
            break;
        case 4 :
            quite = tree;
            break;
        default :
        cout << " That is not a valid input, Quitting program";
        quite = true;
          }
}
return 0;
}
```

Teacher's Signature _____

Enter no. of vertices : 3
is the graph directed ? (y/n) : y
Enter no. of edges : 5
Enter edges (u,v) front (0 based indexing) :

0 1
0 2
1 2
2 0
1 0

Adjancy matrix :

0 1 1
1 0 0
1 1 0

Q.10 Write a program to represent a graph using adjacency matrix.

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

class Graph
{
    private:
     int vertices;
    vector < vector < int >> adj Matrix ;
     bool directed;
    public:
      Graph (int v, bool is directed)
      {
          vertices = v;
        directeded = is directed {
    adj matrix resize ( vertices, vector < int> ( vertices, 0)),
    void add edge ( int u, int v)
    {
    if(u) = vertices ||v >= vertices ||V<0||V<0) {
     cout << " Invalid edge ! " << endl;
        return; }

    adj matrix [u] [v] = 1;
```

Teacher's Signature _____

```
if (!directed)
{
  adj matrix[v][u] = 1; }

}
void display()
 {
  cout << "\n Adjacency matrix : \n";
 for( int i=0; i< vertices; ++i)
 {
  for (int j=0; j< vertices; ++j)
  {
   cout << adj matrix[i][j] << " "; }
   cout << endl;
   }
};
int main() {
     int v, e;
   char graph type;
   cout << " Enter no. of vertices :";
   cin >> v;

 cout << " Is the graph directed ? (y/n) : ";
     cin >> graph type;
```

Teacher's Signature _____

```
bool is directed = 1  graph type == 'Y';
Graph g(v, is Directed);
cout << " enter no of edges :";
  cin >> e;

cout << " Enter edges (u, v) for (0-base indexing) : \n";

For (int i=0; i<e; ++i)
 {
    int u, v;
    cin >> u >> v;
     g.add edge (u, v);
}

    g.display();
     return 0;

}
```

```
Enter the no. of elements : 6
Enter 6 elements :
56
89
45
23
18
57
Enter the element to search : 45
element found at index : 2
```

Q.11 Write a program to implement linear search.

Algorithm

Read the search value to be searched
set i=0
Repeat step 4 untill i>n or arr[i] = search value increment by 1

if i>n:
    Display "Not found"
else
    Display "found"

Program

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

class linear search
{
        int *arr;
        int size;
    public:
        linear Search (int s)
{       size = s;
        arr = new int [size];
3
```

```
void inputArray()
{
    cout << " enter" << size << " elements : " << endl;
  for (int i=0; i<size ; i++)
   {
       cin>> arr[i];
   }
}
int Search (int key)
 {
       for (int i=0; i<size ; i++)
      {
       if(arr[i] == key)
      {   return i;
       }
   }   return -1;
  ~linear Search ()
   {
       delete[] arr;
   }
};  int main() {
       int n, target;
   cout << " enter the no of elements ";
    cin >> n;
   linear search is(n);
   is.inputArray();
```

Teacher's Signature _____

```
cout << " Enter the element to search : ";
 cin >> target ;
int result = is. search (target);
if (result != -1)
{
  cout << " Elements found at index : " << result
       << endl;
}

else { cout << " Element is found : " << endl ;
 }
 return 0;

}
```

```
Enter the no. of elements : 5
Enter 5 elements in ascending order : 34
46
50
65
70
Enter the no of search : 65
Element found at index : 3
```

Q.12 Write a program to implement binary search.

Algorithm

1. Accept the element to be searched
2. set lowerbound = 0
3. set upperbound = n-1
4. set mid (lowerbound + upperbound) /2
5. if arr[mid] = dsired element :
   a) Display "found"
   b) Go to step 10
6. IF dsired element < arr[mid] :
   a) set upperbound = mid - 1
7. IF dsired element > arr[mid] :
   a) set lowerbound = mid + 1
8. IF lowerbound <= upperbound
   a) Go to step 4
9. Display "Not found"
10. Exit.

Teacher's Signature _____

```
#include <iostream>
# include < algorithm>
using namespace std;
class Binary Search :
{
      private :
        int * arr ;
       int  size ;
     public :
Binary Search ( int s)
{       size = s;
        arr = new int [size]
}
void input Array ( )
 {
  cout << " Enter " << size << " sorted elements in
  ascending order";

  for (int i = 0; i < size ; i++)
  {
       cin >> arr[i];
  }
}
```

```
int Search (int key)
{
   int left = 0;
   int right = size - 1;
   while ( left <= right )
   {
      int mid = left + ( right - left ) /2 ;
   if (arr [mid] == key)
   {
      return mid ;
   }
else if(arr [mid] < key)
   {
      left = mid + 1;
   }
else {
      right = mid - 1 ;
   }
}
   return -1;
}
 ~ Binary Search () {
            delete [] arr ;

   }
```

Neel Kamal
PAGE NO.:
DATE / /

```
int main()
{
    int n, target;
    cout << " Enter the no of elements : ";
    cin >> n;
    Binary search bs(n);
    bs.inputArray();
    cout << " Enter the no. to search : ";
    cin >> target;

    int result = bs.search(target);
    if (result != -1)
    Cout << " elements are found at index : " result
                << endl;
else
    Cout << " Not found ";
     return 0;

}
```

Teacher's Signature _____