Shram Sadhana Bombay Trust's

COLLEGE OF ENGINEERING AND TECHNOLOGY,

BAMBHORI POST BOX NO. 94, JALGAON – 425001.(M.S.)

Included under section 2 (f) & 12 (B) of the UGC Act, 1956

NAAC Re-accredited Grade: A (3.14) (2nd Cycle)

ISO 9001: 2008 Certified

Phone No. (0257) 2258393, Fax No. (0257) 2258392

Website- www.sscoetjalgaon.ac.in

Email: sscoetjal@gmail.com

ISO 9001:2008

DEPARTMENT OF COMPUTER APPLICATIONS (M.C.A.)

MCA

SSBT

COET

Laboratory Journal

Class: M.C.A-I

Sem-I

Subject Code: : MCA-414

Subject: LAB on Data Structures and Algorithms

Academic Year:2024-25

University Exam. Seat No:

Name of the Faculty: Asst. Prof. Dhanshree R. Shinde.

Vision of the Institute

Today we carry the flame of quality education, knowledge and progressive technology for global societal development; tomorrow the flame will glow even brighter.

Mission of the Institute

To provide conducive environment for preparing competent, value added and patriotic engineers of integrity of par excellence to meet global standards for societal development.

DEPARTMENT OF COMPUTER APPLICATIONS (M.C.A.)

Vision of the Department

To develop skilled professionals with social values as per industry needs.

Mission of the Department

To provide a student-centred, conducive environment for preparing knowledgeable, skilled and value added professionals to stand out in a competitive world.

Department of Computer Applications (M.C.A.)

Program Outcomes (POs)

PO1: Computational Knowledge: Understand and apply mathematical foundation, computing and domain knowledge for the conceptualization of computing models from defined problems.

PO2: Problem Analysis: Ability to identify, critically analyze and formulate complex computing problems using fundamentals of computer science and application domains.

PO3: **Design / Development of Solutions:** Ability to transform complex business scenarios and contemporary issues into problems, investigate, understand and propose integrated solutions using emerging technologies.

PO4: Conduct Investigations of Complex Computing Problems: Ability to devise and conduct experiments, interpret data and provide well informed conclusions.

PO5: Modern Tool Usage: Ability to select modern computing tools, skills and techniques necessary for innovative software solutions.

PO6: Professional Ethics: Ability to apply and commit professional ethics and cyber regulations in a global economic environment.

PO7: Life-long Learning: Recognize the need for and develop the ability to engage in continuous learning as a Computing professional.

PO8: Project Management and Finance: Ability to understand, management and computing principles with computing knowledge to manage projects in multidisciplinary environments.

PO9: Communication Efficacy: Communicate effectively with the computing community as well as society by being able to comprehend effective documentations and presentations.

PO10: Societal & Environmental Concern: Ability to recognize economic, environmental, social, health, legal, ethical issues involved in the use of computer technology and other consequential responsibilities relevant to professional practice.

PO11: Individual and Team Work: Ability to work as a member or leader in diverse teams in multidisciplinary environment.

PO12: Innovation and Entrepreneurship: Identify opportunities, entrepreneurship vision and use of innovative ideas to create value and wealth for the betterment of the individual and society.

Department of Computer Applications (M.C.A.)

Program Educational Objectives (PEOs)

PEO 1: Core Knowledge -The computing professional graduate will have the knowledge of basic science and Engineering skills, Humanities, social science, management and conceptual and practical understanding of core computer applications area with project development.

PEO 2: Employment/ Continuing Education – The computing professional graduate will have the knowledge of Industry-based technical skills to succeed in entry level professional position at various industries as well as in academics.

PEO 3: Professional Competency - The computing professional graduate will have the ability to communicate effectively in English, to accumulate and disseminate the knowledge and to work effectively in a team with a sense of social awareness.

Department of Computer Applications (M.C.A.)

Program Specific Outcomes (PSOs)

PSO 1: The computing professional graduate will be able to apply knowledge of computer science in practice to identify, critically analyze, formulate and develop computer applications using modern computing tools and techniques and will use these tools with dexterity.

PSO 2: The computing professional graduate will be able to design computing systems to meet desired needs within realistic constraints such as safety, security and applicability. These systems will function professionally with ethical responsibility as an individual as well as in multidisciplinary teams with positive attitude.

PSO 3: The computing professional graduate will be able to appreciate the importance of goal setting and recognize the need for life-long learning with good communication skills.

Department of Computer Applications (M.C.A.)

MCA-414 LAB on Data Structures and Algorithms

M.C.A-I Sem-I

Course Outcomes:

CO1: Ability to analyze the time and space complexities of algorithms.

CO2: Understand the difference between structured data and data structure.

CO3: Choose the appropriate data structure and algorithm design method for a specified application.

CO4: Ability to design programs using a variety of data structures such as stacks, queues, binary trees, search trees, etc.

CO-PO and PSO Mappings

Course Outcomes	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PS O1	PS O2	PS O3
CO1:	3	3		3									3	3	
CO2:	3	2											3	2	
CO3:	3	3	3		3							2	3	3	
CO4:	3	3	3	3	3						3	3	3	3	
CO5:	**3**	**2.7**	**3**	**3**	**3**	**0**	**0**	**0**	**0**		**3**	**2.5**	**3**	**2.7**	**0**

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 01

Date of Performance: **Date of Completion:**

Title: To implement a program in cpp for Array.

Code:

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

int main() {
   int A[100];
   int n, sum = 0;

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

for (int i = 0; i < n; i++) {
cout << "Enter array element: ";
 cin >> A[i];
 }

  for (int i = 0; i < n; i++) {
 sum += A[i];
 }

cout << "Summation of Array is = " << sum << endl;

 return 0;
}
```

Output:

```
Enter size of array : 5
Enter Array elements:
10
20
30
40
50
Summation of Array is = 150
```

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R.Shinde.**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 02

Date of Performance: **Date of Completion:**

Title: To implement a program in cpp for Multidimensional Arrays

---

Code:

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

 int main() {
   int rows, cols;

  // Input the dimensions of the matrix
 cout << "Enter the number of rows and columns: ";
 cin >> rows >> cols;

int arr1[10][10], arr2[10][10], sum[10][10];

  // Input elements of the first matrix
    cout << "Enter elements of first matrix:\n";
 for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
      cin >> arr1[i][j];
 }
}

  // Input elements of the second matrix
 cout << "Enter elements of second matrix:\n";
 for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
  cin >> arr2[i][j];
 }
}

  // Adding the two matrices
    for (int i = 0; i < rows; i++) {
```

```
 for (int j = 0; j < cols; j++) {
      sum[i][j] = arr1[i][j] + arr2[i][j];
     }

}

    // Display the sum matrix
    cout << "Sum of the matrices:\n";
   for (int i = 0; i < rows; i++) {
       for (int j = 0; j < cols; j++) {
        cout << sum[i][j] << " ";
    }
     cout << endl;
}

 return 0;
  }
```

Output:

Enter the number of rows and columns: 2 2
Enter elements of first matrix:
1 2
2 1
Enter elements of second matrix:
2 2
1 1
Sum of the matrices:
3 4
3 2

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R.Shinde.**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 03

Date of Performance: **Date of Completion:**

Title: To implement a program in cpp for Matrices.

Code:

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

int main() {
int rows, cols;

// Input the dimensions of the matrix
cout << "Enter the number of rows and columns: ";
cin >> rows >> cols;

int arr1[10][10], arr2[10][10], sum[10][10];

// Input elements of the first matrix
cout << "Enter elements of first matrix:\n";
for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
cin >> arr1[i][j];
}
}

// Input elements of the second matrix
cout << "Enter elements of second matrix:\n";
for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
cin >> arr2[i][j];
}
}

// Adding the two matrices
for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
sum[i][j] = arr1[i][j] + arr2[i][j];
}
```

```
    }

// Display the sum matrix
 cout << "Sum of the matrices:\n";
   for (int i = 0; i < rows; i++) {
     for (int j = 0; j < cols; j++) {
       cout << sum[i][j] << " ";
  }
    cout << endl;
    }

 return 0;
}
```

Output:

```
Enter the number of rows and columns: 2 2
Enter elements of first matrix:
2 1
2 1
Enter elements of second matrix:
2 2
1 1
Sum of the matrices:
4 3
3 2
```

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R. Shinde.**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 04

Date of Performance: **Date of Completion:**

Title: To implement a program in cpp for Stack .

Code:

```
#include <iostream>
#include <cstdlib>
class stack{
private:
static const int MAX_SIZE = 10;
int s[MAX_SIZE];
int top;
int ele;
int i;

public:
stack() {
top = -1;
}
void push();
void display();
void pop();
void peep();
void change();
};
void stack::push() {
                if (top>=MAX_SIZE-1)
                std::cout << "\nstack is overflow:";
        else {
                        std::cout << "\nEnter element:"; std::cin >> ele;
                        top++; s[top] = ele;
        }
   }
   void stack::display() {
                    if (top == -1)
                    std::cout << "\nstack is Empty";
```

```
else {

  std::cout << "\nElements in stack are:\n";
          for (i = top; i >= 0; i--)
          std::cout << s[i] << "\t";
}

}
void stack::pop() {
                 if (top == -1)
                std::cout << "\nUnderflow";
else {
           std::cout << "\npop ele is " << s[top]; top--;
}
}

void stack::peep() {
                std::cout << "\nEnter position:"; std::cin >> i;
                 if ((top - i + 1) < 0)
                std::cout << "\nUnderflow"; else
                 std::cout << "\nPeep ele is " << s[top - i + 1];
}

void stack::change() {
                    std::cout << "\nEnter position: "; std::cin >> i;
                    if ((top - i + 1) < 0)
                    std::cout << "\nUnderflow";
else {
int n;
std::cout << "\nEnter element:"; std::cin >> n;
s[top - i + 1] = n;
}
}
int main() { stack s; int ch; int n;
std::cout << "Enter size of stack: "; std::cin >> n;
while (true) {
std::cout << "\n1. Push 2. Display 3. Pop 4. Peep 5. Change 6. Exit\n"; std::cout <<
"\nEnter choice: ";
std::cin >> ch;
switch (ch) {
```

```
    case 1:

    s.push();
    break;

        case 2:
        s.display();
        break;
        case 3:
        s.pop();
        break;
        case 4:
        s.peep();
        break;
        case 5:
        s.change();
        break;
        case 6:
        exit(0);
        default:
        std::cout << "Invalid choice!";

        }

        }
        return 0;

}
```

Output:

Enter size of stack: 5

1. Push 2. Display 3. Pop 4. Peep 5. Change 6. Exit

Enter choice: 1

Enter element:1

1. Push 2. Display 3. Pop 4. Peep 5. Change 6. Exit

Enter choice: 1

Enter element:2

1. Push 2. Display 3. Pop 4. Peep 5. Change 6. Exit

Enter choice: 1

Enter element:3

1. Push 2. Display 3. Pop 4. Peep 5. Change 6. Exit

Enter choice: 1

Enter element:4

1. Push 2. Display 3. Pop 4. Peep 5. Change 6. Exit

Enter choice: 2

Elements in stack are:

4 3 2 1

1. Push 2. Display 3. Pop 4. Peep 5. Change 6. Exit

Enter choice: 3

pop ele is 4
1. Push 2. Display 3. Pop 4. Peep 5. Change 6. Exit

Enter choice: 4

Enter position:2

Peep ele is 2
1. Push 2. Display 3. Pop 4. Peep 5. Change 6. Exit

Enter choice: 5

Enter position: 1

Enter element:3

1. Push 2. Display 3. Pop 4. Peep 5. Change 6. Exit

Enter choice: 2

Elements in stack are:
3 2 1
1. Push 2. Display 3. Pop 4. Peep 5. Change 6. Exit

Enter choice:

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R. Shinde.**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 05

Date of Performance: **Date of Completion:**

Title: To implement a program in cpp for Infix to Postfix .

Code:

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

using namespace std;

// Function to return precedence of operators
int precedence(char op) {
    if (op == '^') return 3;
    if (op == '*' || op == '/') return 2;
    if (op == '+' || op == '-') return 1;
    return 0;
}

// Function to check if the character is an operator
bool isOperator(char c) {
    return (c == '+' || c == '-' || c == '*' || c == '/' || c == '^');
}

// Function to convert infix expression to postfix
string infixToPostfix(string infix) {
    stack<char> s;  // Stack to hold operators and parentheses
    string postfix; // Resulting postfix expression

    for (int i = 0; i < infix.length(); i++) {
        char ch = infix[i];

        // If the character is an operand (number or letter), add it to postfix
        if (isalnum(ch)) {
            postfix += ch;
        }

    // If the character is '(', push it to the stack
```

```
        else if (ch == '(') {
            s.push(ch);
        }
        // If the character is ')', pop from stack until '(' is found
        else if (ch == ')') {
            while (!s.empty() && s.top() != '(') {
                postfix += s.top();
                s.pop();
            }
            s.pop(); // Remove '(' from the stack
        }
        // If the character is an operator
        else if (isOperator(ch)) {
            while (!s.empty() && precedence(s.top()) >= precedence(ch)) {
                postfix += s.top();
                s.pop();
            }
            s.push(ch);
        }
}

    // Pop remaining operators from the stack
    while (!s.empty()) {
        postfix += s.top();
        s.pop();
    }

    return postfix;
}

int main() {
    string infix;

    // Get user input for infix expression
    cout << "Enter an infix expression: ";
    cin >> infix;

    // Convert infix to postfix
    string postfix = infixToPostfix(infix);
```

```
 // Output the postfix expression
    cout << "Postfix Expression: " << postfix << endl;

    return 0;
}
```

Output:

Enter an infix expression: +ABC

Postfix Expression: ABC+

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R. Shinde.**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon
Department of Computer Applications
Practical: 06

Date of Performance: **Date of Completion:**

Title: **To implement a program in cpp for Infix to Prefix.**

Code:

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

int main() {
   char p[30], temp[50], opr[10], op1[10], op2[10];
   stack<string> stk;

   // Take input postfix expression
   cout << "\nEnter the postfix expression: ";
   cin >> p;

   int i = 0;

   // Process each character of the postfix expression
   while(p[i] != '\0') {
      // If the character is an operator
      if(p[i] == '+' || p[i] == '-' || p[i] == '*' || p[i] == '/' || p[i] == '^') {

         // Pop the two operands from the stack
         op2[0] = '\0'; op1[0] = '\0'; opr[0] = p[i];
         opr[1] = '\0';  // Set operator

         // Pop two operands from the stack
         op2 = stk.top(); stk.pop();
         op1 = stk.top(); stk.pop();
```

```
            // Create the infix expression in the form (op1 operator op2)
            strcpy(temp, "(");
            strcat(temp, op1.c_str());  // Append op1 to temp
            strcat(temp, opr);          // Append operator
            strcat(temp, op2.c_str());  // Append op2
            strcat(temp, ")");

            // Push the result back into the stack
            stk.push(temp);
        }
        else {
            // If it's an operand, push it to the stack
            temp[0] = p[i];
            temp[1] = '\0';
            stk.push(temp);
        }
        i++;
    }

    // The top of the stack will contain the infix expression
    cout << "\nInfix expression is: " << stk.top() << endl;

    return 0;
}
```

Output:

Enter the postfix expression: ab+c*

Infix expression is: (a+b)*c

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R.Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 07

Date of Performance: **Date of Completion:**

Title:To implement a program in cpp for Queue using array.

Code:

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

int m;  // Size of the queue

class queue {
   int f, r, q[10], n;  // f = front, r = rear, q = array for queue, n = element to insert

public:
   // Constructor to initialize front and rear to 0
   queue() {
      f = r = 0;
   }

   void insert();  // Insert function
   void del();     // Delete function
   void dis();     // Display function
};

// Function to insert an element in the queue
void queue::insert() {
   if (r == m) {
      cout << "\nOverflow (Queue is full)";
   } else {
      cout << "\nEnter Element in Queue = ";
      cin >> n;  // Read the element to be inserted
      if (f == 0) {
         f = 1;  // Set front to 1 if it's the first insertion
      }
```

```
    r++;  // Move the rear to the next position

 q[r] = n;  // Insert the element at the rear

}
}

// Function to delete an element from the queue
void queue::del() {
    if (f == 0) {
        cout << "\nUnderflow (Queue is empty)";
    } else {
        int n = q[f];  // Get the element at the front
        if (f == r) {
            // If front equals rear, the queue becomes empty after deletion
            f = r = 0;
        } else {
            f++;  // Move the front to the next position
        }
        cout << "\nDeleted element is " << n;
    }
}

// Function to display the elements in the queue
void queue::dis() {
    if (f == 0) {
        cout << "\nUnderflow (Queue is empty)";
    } else {
        cout << "\nElements in queue are: ";
        for (int i = f; i <= r; i++) {
            cout << q[i] << "\t";  // Display each element from front to rear
        }
    }
}

// Main function
int main() {
    queue q;  // Create an object of the queue class
    int ch;   // Variable to store user's choice
```

```
 cout << "Enter size of queue: ";
   cin >> m;  // Read the maximum size of the queue

   cout << "\n1. Insert\n2. Display\n3. Delete\n4. Exit\n";


    while (ch != 4) {
      cout << "\nEnter choice: ";
      cin >> ch;  // Read user's choice

      switch (ch) {
         case 1:
            q.insert();  // Call insert function
            break;
         case 2:
            q.dis();     // Call display function
            break;
         case 3:
            q.del();     // Call delete function
            break;
         case 4:
            exit(0);     // Exit the program
         default:
            cout << "\nInvalid choice!";
      }
   }

   return 0;
}
```

Output:

```
Enter size of queue: 5

1. Insert
2. Display
3. Delete
4. Exit


Enter choice: 1

Enter Element in Queue = 1

Enter choice: 1

Enter Element in Queue = 2

Enter choice: 1

Enter Element in Queue = 3

Enter choice: 1

Enter Element in Queue = 4

Enter choice: 1

Enter Element in Queue = 5

Enter choice: 2

Elements in queue are: 1       2       3       4       5
Enter choice: 3


Deleted element is 1
Enter choice:
```

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R.Shinde.**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 08

Date of Performance: **Date of Completion:**

Title: To implement a program in CPP for Deques.

Code:

```
#include <iostream>
#define SIZE 5

using namespace std;

int deque[SIZE];
int f = -1, r = -1;

// insert_front function will insert the value from the front
void insert_front(int x) {
   if ((f == 0 && r == SIZE - 1) || (f == r + 1)) {
      cout << "Overflow" << endl;
   } else if (f == -1 && r == -1) {
      f = r = 0;
      deque[f] = x;
   } else if (f == 0) {
      f = SIZE - 1;
      deque[f] = x;
   } else {
      f = f - 1;
      deque[f] = x;
   }
}

// insert_rear function will insert the value from the rear
void insert_rear(int x) {
   if ((f == 0 && r == SIZE - 1) || (f == r + 1)) {
      cout << "Overflow" << endl;
   } else if (f == -1 && r == -1) {
      r = 0;
      deque[r] = x;
   } else if (r == SIZE - 1) {
      r = 0;
      deque[r] = x;
```

```
  } else {
        r++;
        deque[r] = x;
    }
}

// display function prints all the values of deque
void display() {
    if (f == -1 && r == -1) {
        cout << "Deque is empty" << endl;
        return;
    }

    int i = f;
    cout << "Elements in deque are: ";
    while (i != r) {
        cout << deque[i] << " ";
        i = (i + 1) % SIZE;
    }
    cout << deque[r] << endl;
}

// getfront function retrieves the first value of the deque
void getfront() {
    if (f == -1 && r == -1) {
        cout << "Deque is empty" << endl;
    } else {
        cout << "The value at the front is: " << deque[f] << endl;
    }
}

// getrear function retrieves the last value of the deque
void getrear() {
    if (f == -1 && r == -1) {
        cout << "Deque is empty" << endl;
    } else {
        cout << "The value at the rear is: " << deque[r] << endl;
    }
}

// delete_front() function deletes the element from the front
void delete_front() {
    if (f == -1 && r == -1) {
        cout << "Deque is empty" << endl;
```

```
} else if (f == r) {
        cout << "The deleted element is: " << deque[f] << endl;
        f = r = -1;
    } else if (f == SIZE - 1) {
        cout << "The deleted element is: " << deque[f] << endl;
        f = 0;
    } else {
        cout << "The deleted element is: " << deque[f] << endl;
        f = f + 1;
    }
}

// delete_rear() function deletes the element from the rear
void delete_rear() {
    if (f == -1 && r == -1) {
        cout << "Deque is empty" << endl;
    } else if (f == r) {
        cout << "The deleted element is: " << deque[r] << endl;
        f = r = -1;
    } else if (r == 0) {
        cout << "The deleted element is: " << deque[r] << endl;
        r = SIZE - 1;
    } else {
        cout << "The deleted element is: " << deque[r] << endl;
        r = r - 1;
    }
}

int main() {
    insert_front(20);
    insert_front(10);
    insert_rear(30);
    insert_rear(50);
    insert_rear(80);

    display(); // Display the values of deque
    getfront(); // Retrieve the value at front-end
    getrear();  // Retrieve the value at rear-end
    delete_front();
    delete_rear();
    display(); // Display the values after deletion

    return 0;
}
```

Output:

Elements in deque are: 10 20 30 50 80
The value at the front is: 10
The value at the rear is: 80
The deleted element is: 10
The deleted element is: 80
Elements in deque are: 20 30 50

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R. Shinde.**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 09

Date of Performance: **Date of Completion:**

Title : To implement a program in CPP for Linear(Single) Linked List.

Code:

```
#include<iostream>
#include<conio.h>
#include<process.h>
using namespace std;
class node {
int info, item, s; node *link;
public:

void insert();
void dis();
void del();
void search();
void sum();
};

node *move, *start = NULL, *temp;
void node::insert() {
cout << "\nEnter the item:";
cin >> item;
node *node1 = new node;
node1->link = NULL;
node1->info = item;
```

```
if (start == NULL) start = node1;
else {

move = start;

while (move->link != NULL) move = move->link;
move->link = node1;

}

}

void node::dis() { node *x;
x = start;

cout << "\n Elements in LL are:";
while (x != NULL) {
cout << "\t" << x->info; x = x->link;
}

}

void node::sum() {
node *x;
x = start;
s = 0;
while (x != NULL) {
s = s + x->info;
x = x->link;
```

```
}

cout << "\nSum of node is" << s;

}

void node::del() {
temp = start;
if (temp != NULL) {
 temp = temp->link;
cout << "\nDeleted node is" << start->info; delete start;
start = temp;

} else

cout << "\n List is empty:";

}

void node::search() {
int c = 0, f = 0, d;
cout << "\nEnter item"; cin >> item;
temp = start;

while (temp != NULL) {
c++;
if (temp->info == item) {
f = 1;
d = c;
```

```
break;
}

temp = temp->link;

}

if (f == 1)

cout << "\nElement is found at position " << d;
else
cout << "\nElement is not found";

}

int main() { node n; int ch;
cout << "\n1.Insert 2.Display 3. Delete 4.Search 5.Sum 6.Exit\n";
do {
cout << "\nEnter choice"; cin >> ch;
switch (ch) {

case 1:

n.insert();
break;
case 2:

n.dis();
 break;
```

```
case 3:

n.del();
 break;
case 4:

n.search();
break;
case 5:

n.sum();
break;
case 6:

return 0;

}

}
while (ch != 6);
getch();
return 0;

}
```

Output:

1.Insert 2.Display 3. Delete 4.Search 5.Sum 6.Exit

Enter choice1

Enter the item:1

Enter choice1

Enter the item:2

Enter choice1

Enter the item:3

Enter choice1

Enter the item:4

Enter choice1

Enter the item:5

Enter choice2

Elements in LL are: 1 2 3 4 5
Enter choice3

Deleted node is1
Enter choice4

Enter item2

Element is found at position 1
Enter choice5

Sum of node is14
Enter choice

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R. Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 10

Date of Performance: **Date of Completion:**

Title : To implement a program in CPP for Circular Linked List.

Code:

```
#include<iostream>
 #include<conio.h>
 #include<cstdlib>
 using namespace
 std;
 class node {
 int info,c,i;
 node *link;
 public:

 node() {
 c=0;
 }

 void insert();
 void display();
 void del();
 };

 node
 *start=NULL,
 *temp=NULL,
 *move=NULL,
 *temp1=NULL;
 void node::insert()
 {
 int item;
```

```
node *p=new
node;
cout<<"\nEnter
Element:";
cin>>item;


p->info=item;
p->link=NULL;
if(start==NULL) {
start=p;
p->link=start;
c++;
}

else {

temp=start;

while(temp-
>link!=start)
temp=temp->link;
temp->link=p;
p->link=start;
c++;
}

}

void
node::display() {
if(start==NULL) {
cout<<"\n LL
empty";
return;
}

node *temp;
temp=start;
```

```
move=start->link;
cout<<temp->info;
while(move!=start)
{
cout<<"-
>"<<move->info;
move=move->link;
}

cout<<"\n Number
of


nodes in CLL are
:"<<c;

}

void node::del() {
int pos;
cout<<"\nEnter
Position:";
cin>>pos;
if(c==1) {

start=NULL;

}

if(start==NULL) {
cout<<"\n LL
Empty:";
return;
}

if(pos>c || pos<1) {
cout<<"\nInvalid
Position";
return;
```

```
}

if(pos==1) {
temp=start;
while(temp-
>link!=start)
temp=temp->link;
temp1=start;
start=start->link;
temp->link=start;
cout<<"\nDeleted
Element is
"<<temp1->info;
delete temp1;


c--;

}

else {

temp=start; i=1;
while(i<pos-1) {
temp=temp->link;
i++;
}

temp1=temp->link;

temp->link=temp1-
>link;

cout<<"\nDeleted
element
is"<<temp1->info;
delete temp1;
c--;
```

```
}

}



int main() {
node n;
int ch;
cout<<"\n 1.Insert
2.Display 3.Delete
4.Exit";
while(ch!=4) {
cout<<"\n Enter
Choice";
cin>>ch;
switch(ch) {


case 1: n.insert();
break;
case 2: n.display();
break;
case 3: n.del();
break;
case 4:
exit(0);
}

}

getch();

return 0;
}
```

Output:

1. Insert
2. Display
3. Delete
4. Exit

Enter Choice: 1
Enter Element:
10

Enter Choice: 1
Enter Element:
20

Enter Choice: 1
Enter Element:
30

Enter Choice: 2
10->20->30
Number of
nodes in CLL
are: 3

Enter Choice: 3
Enter Position:
2
Deleted
Element is 20

Enter Choice: 2
10->30
Number of
nodes in CLL
are: 2
Enter Choice: 4
Exit

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R. Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 11

Date of Performance: **Date of Completion:**

Title : To implement a program in CPP for Doubly Linked List.

Code:

```
#include<iostream>
#include<conio.h>
#include<cstdlib>
using namespace std;
class node {
int info,c,j;

node *left,*right;
public:
void insert();
void display();
void del();
};

node
*start=NULL,*temp=
NULL,*move=NULL,
*temp1=NULL;
void node::insert() {
int item;
```

```
node *p=new node;
cout<<"\nEnter
element:";
cin>>item;
p->info=item;
p->left=NULL;
p->right=NULL;
if(start==NULL) {
start=p;
return;
}

else {

temp=start;

while(temp-
>right!=NULL)

temp=temp->right;
temp->right=p;
p->left=start;

}
```

```
}

void node::display() {
move=start;
if(move==NULL) {
cout<<"\n LL Empty:";
return;
}

else {

cout<<"\n node in DLL
are :";
while(move!=NULL) {
cout<<move-
>info<<"\t";
move=move->right;
}

}

}

void node::del() {
```

```
if(start==NULL) {
cout<<"\n LL Empty:";
return;
}

temp=start; start=temp->right;
if(start != NULL)
start->left=NULL;
temp->right=NULL;

cout<<"\n deleted element is"<<temp->info;
delete temp;
}

int main() {
node n;
int ch;
cout<<"\n1. Insert 2. Display 3. Delete 4. Exit";
while(ch!=4) {
cout<<"\nEnter choice";
```

```
cin>>ch;
switch(ch) {

case 1: n.insert();
break;
case 2: n.display();
break;
case 3: n.del();
break;
case 4:
exit(0);
}

}

getch();
return 0;
}
```

Output :

```
1. Insert 2. Display 3. Delete 4. Exit
Enter choice: 1
Enter element: 10

Enter choice: 1
Enter element: 20

Enter choice: 1
Enter element: 30

Enter choice: 2
Nodes in DLL are: 10    20    30

Enter choice: 3
Deleted element is: 10

Enter choice: 2
Nodes in DLL are: 20    30

Enter choice: 4
```

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R. Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 12

Date of Performance: **Date of Completion:**

Title: To implement the program in CPP for Polynomial Addition.

Code:

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

   int main() {
   int a[10], b[10], c[10];
   int m, n, k = 0, k1, i, j;

   cout << "\n\tPolynomial Addition\n";
   cout << "\t==================\n";

   // Input for the first polynomial
   cout << "\n\tEnter the number of terms of the polynomial: ";
   cin >> m;
   cout << "\n\tEnter the degrees and coefficients: ";
   for (i = 0; i < 2 * m; i++) {
      cin >> a[i];
   }

   // Display the first polynomial
   cout << "\n\tFirst polynomial is: ";
   k1 = 0;
   if (a[k1 + 1] == 1) {
      cout << "x^" << a[k1];
   } else {
      cout << a[k1 + 1] << "x^" << a[k1];
   }
   k1 += 2;
   while (k1 < 2 * m) {
      cout << "+" << a[k1 + 1] << "x^" << a[k1];
      k1 += 2;
}
```

```
// Input for the second polynomial
 cout << "\n\n\tEnter the number of terms of the second polynomial: ";
 cin >> n;
 cout << "\n\tEnter the degrees and coefficients: ";
  for (j = 0; j < 2 * n; j++) {
  cin >> b[j];
  }

   // Display the second polynomial
 cout << "\n\tSecond polynomial is: ";
  k1 = 0;
 if (b[k1 + 1] == 1) {
    cout << "x^" << b[k1];
 } else {
   cout << b[k1 + 1] << "x^" << b[k1];
           }
  k1 += 2;
 while (k1 < 2 * n) {
  cout << "+" << b[k1 + 1] << "x^" << b[k1];
  k1 += 2;
  }

   // Add the two polynomials
  i = 0;
   j = 0;
 while (m > 0 && n > 0) {
  if (a[i] == b[j]) {
  c[k + 1] = a[i + 1] + b[j + 1];
  c[k] = a[i];
   m--;
  n--;
  i += 2;
 j += 2;
 } else if (a[i] > b[j]) {
   c[k + 1] = a[i + 1];
 c[k] = a[i];
           m--;
  i += 2;
 } else {
  c[k + 1] = b[j + 1];
 c[k] = b[j];
   n--;
  j += 2;
```

```
 }
 k += 2;
}

  // Add remaining terms of the first polynomial
while (m > 0) {
  c[k + 1] = a[i + 1];
 c[k] = a[i];
k += 2;
 i += 2;
 m--;
}

 // Add remaining terms of the second polynomial
 while (n > 0) {
 c[k + 1] = b[j + 1];
 c[k] = b[j];
  k += 2;
 j += 2;
  n--;
  }

 // Display the resulting polynomial
cout << "\n\n\tSum of the two polynomials is: ";
k1 = 0;
 if (c[k1 + 1] == 1) {
 cout << "x^" << c[k1];
 } else {
     cout << c[k1 + 1] << "x^" << c[k1];
}
  k1 += 2;
  while (k1 < k) {
if (c[k1 + 1] == 1) {
   cout << "+x^" << c[k1];
  } else {
   cout << "+" << c[k1 + 1] << "x^" << c[k1];
}
  k1 += 2;
 }

cout << endl;
return 0;
}
```

Output:

Enter the number of terms of the polynomial: 2
Enter the degrees and coefficients: 2 3 0 4

First polynomial is: 3x^2+4x^0

Enter the number of terms of the second polynomial: 2
Enter the degrees and coefficients: 3 1 2 2

Second polynomial is: 1x^3+2x^2

Sum of the two polynomials is: 1x^3+5x^2+4x^0

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R .Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 13

Date of Performance: **Date of Completion:**

Title: To implement the program in CPP for Polynomial Addition sing Linked List.

Code:

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

// Node structure containing power and coefficient of variable
struct Node {
   int coeff;
   int pow;
   Node* next;
};

// Function to create a new node
void create_node(int coeff, int pow, Node** temp) {
   Node *r, *z;
   z = *temp;
   if (z == nullptr) {
      r = new Node();
      r->coeff = coeff;
      r->pow = pow;
      *temp = r;
      r->next = new Node();
      r = r->next;
      r->next = nullptr;
   } else {
      r->coeff = coeff;
      r->pow = pow;
      r->next = new Node();
      r = r->next;
      r->next = nullptr;
   }
}
```

```
// Function to add two polynomial numbers
void polyadd(Node* poly1, Node* poly2, Node* poly) {
    while (poly1->next != nullptr && poly2->next != nullptr) {
        // If power of 1st polynomial is greater than 2nd
        if (poly1->pow > poly2->pow) {
            poly->pow = poly1->pow;
            poly->coeff = poly1->coeff;
            poly1 = poly1->next;
        }
 // If power of 2nd polynomial is greater than 1st
          else if (poly1->pow < poly2->pow) {
poly->pow = poly2->pow;
  poly->coeff = poly2->coeff;
poly2 = poly2->next;
          }
// If power of both polynomials is the same
else {
poly->pow = poly1->pow;
poly->coeff = poly1->coeff + poly2->coeff;
poly1 = poly1->next;
poly2 = poly2->next;
}
// Dynamically create a new node
poly->next = new Node();
poly = poly->next;
poly->next = nullptr;
 }

 // Add remaining terms of the first polynomial
while (poly1->next != nullptr) {
  poly->pow = poly1->pow;
poly->coeff = poly1->coeff;
  poly1 = poly1->next;
poly->next = new Node();
 poly = poly->next;
poly->next = nullptr;
 }

          // Add remaining terms of the second polynomial
while (poly2->next != nullptr) {
poly->pow = poly2->pow;
poly->coeff = poly2->coeff;
```

```
poly2 = poly2->next;

poly->next = new Node();
        poly = poly->next;
        poly->next = nullptr;
    }
}

// Function to display the polynomial
void show(Node* node) {
    while (node->next != nullptr) {
        cout << node->coeff << "x^" << node->pow;
        node = node->next;
        if (node->coeff >= 0 && node->next != nullptr) {
            cout << "+";
        }
    }
}

int main() {
    Node *poly1 = nullptr, *poly2 = nullptr, *poly = nullptr;

    // Create first polynomial: 5x^2 + 4x^1 + 2x^0
    create_node(5, 2, &poly1);
    create_node(4, 1, &poly1);
    create_node(2, 0, &poly1);

    // Create second polynomial: -5x^1 - 5x^0
    create_node(-5, 1, &poly2);
    create_node(-5, 0, &poly2);

    cout << "1st Polynomial: ";
    show(poly1);

    cout << "\n2nd Polynomial: ";
    show(poly2);

    // Create result polynomial
    poly = new Node();

    // Add the two polynomials
    polyadd(poly1, poly2, poly);

    // Display the result
```

```
 cout << "\nAdded Polynomial: ";
 show(poly);
return 0;
}
```

Output:

1st Polynomial: 5x^2+4x^1+2x^0
2nd Polynomial: -5x^1-5x^0
Added Polynomial: 5x^2-3x^0

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R. Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 14

Date of Performance: **Date of Completion:**

Title:To implement a program in cpp for Binary Search Tree.

Code:

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

// Definition of the BST node
struct Node {
    int data;
    Node *left_child;
    Node *right_child;

    Node(int value) {
        data = value;
        left_child = nullptr;
        right_child = nullptr;
    }
};

// Function to create a new node
Node* new_node(int x) {
    return new Node(x);
}

// Function to search for a value in the BST
Node* search(Node* root, int x) {
    if (root == nullptr || root->data == x)
        return root;

    if (x > root->data)
        return search(root->right_child, x);
    else
        return search(root->left_child, x);
}
```

```
// Function to insert a value into the BST
Node* insert(Node* root, int x) {
    if (root == nullptr)
        return new_node(x);

    if (x > root->data)
        root->right_child = insert(root->right_child, x);
    else
        root->left_child = insert(root->left_child, x);

    return root;
}

// Function to find the minimum value in the BST
Node* find_minimum(Node* root) {
    if (root == nullptr)
        return nullptr;

    if (root->left_child != nullptr)
        return find_minimum(root->left_child);

    return root;
}

// Function to delete a value from the BST
Node* delete_node(Node* root, int x) {
    if (root == nullptr)
        return nullptr;

    if (x > root->data)
        root->right_child = delete_node(root->right_child, x);
    else if (x < root->data)
        root->left_child = delete_node(root->left_child, x);
    else {
        // Node with no children
        if (root->left_child == nullptr && root->right_child == nullptr) {
            delete root;
            return nullptr;
        }
        // Node with one child
        else if (root->left_child == nullptr || root->right_child == nullptr) {
            Node* temp = (root->left_child == nullptr) ? root->right_child : root-
>left_child;
```

```
  delete root;
return temp;
}
          // Node with two children
  else {
 Node* temp = find_minimum(root->right_child);
root->data = temp->data;
 root->right_child = delete_node(root->right_child, temp->data);
}
}
   return root;
}

// Function for in-order traversal
void inorder(Node* root) {
   if (root != nullptr) {
      inorder(root->left_child);
      cout << root->data << " ";
      inorder(root->right_child);
   }
}

int main() {
   Node* root = new_node(20);
   insert(root, 5);
   insert(root, 1);
   insert(root, 15);
   insert(root, 9);
   insert(root, 7);
   insert(root, 12);
   insert(root, 30);
   insert(root, 25);
   insert(root, 40);
   insert(root, 45);
   insert(root, 42);

   cout << "In-order traversal before deletion: ";
   inorder(root);
   cout << endl;

   // Deleting nodes
   root = delete_node(root, 1);
   root = delete_node(root, 40);
```

```
 root = delete_node(root, 45);
    root = delete_node(root, 9);

    cout << "In-order traversal after deletion: ";
    inorder(root);
    cout << endl;

    return 0;
}
```

Output:

1 5 7 9 12 15 20 25 30 40 42 45
5 7 12 15 20 25 30 42

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R.Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 15

Date of Performance: **Date of Completion:**

Title: To implement a program in cpp for In-order, Pre-order, Post-order Traversals.

Code:

```
include <iostream>
using namespace std;
struct ver {
int data;

ver *left, *right;
};
class tree {
public:
ver* create(int, ver*);
void in(ver*);
void post(ver*);
void pre(ver*);
};


ver* tree::create(int c, ver* node) {
if (node == NULL) {
node = new ver;
node->data = c;
node->left = NULL;
node->right = NULL;
return node;
}

else {
if (c < node->data)
node->left = create(c, node->left);
else
node->right = create(c, node->right);
return node;

}
```

```
}


void tree::in(ver* node) {
if (node) {
in(node->left);
cout << node->data << "\t";
in(node->right);
}
}


void tree::pre(ver* node) {
if (node) {
cout << node->data << "\t";
pre(node->left);
pre(node->right);
}

}


void tree::post(ver* node) {
if (node) {
post(node->left);
post(node->right);
cout << node->data << "\t";
}
}


int main() {
tree t;
ver* r = NULL;
int n, ch;


cout << "\n 1: Insert 2: Inorder 3: Preorder 4: Postorder 5: Exit :\n";
while (ch != 5) {
cout << "\nEnter Choice:";
cin >> ch;
switch (ch) {
```

```
case 1:
cout << "\nEnter Node:";
cin >> n;

r = t.create(n, r);
break;
case 2:
cout << "\nInorder Traversal:";
t.in(r);
break;
case 3:
cout << "\nPreorder Traversal:";
t.pre(r);
break;
case 4:
cout << "\nPostorder Traversal:";
t.post(r);
break;
case 5:
return 0;

}

}

}
```

Output:

```
1: Insert  2: Inorder  3: Preorder  4: Postorder  5: Exit :

Enter Choice: 1
Enter Node: 50

Enter Choice: 1
Enter Node: 30

Enter Choice: 1
Enter Node: 70

Enter Choice: 1
Enter Node: 20

Enter Choice: 1
Enter Node: 40

Enter Choice: 1
Enter Node: 60

Enter Choice: 1
Enter Node: 80

Enter Choice: 2
Inorder Traversal:
20    30    40    50    60    70    80

Enter Choice: 3
Preorder Traversal:
50    30    20    40    70    60    80

Enter Choice: 4
Postorder Traversal:
20    40    30    60    80    70    50

Enter Choice: 5
```

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R.Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 16

Date of Performance: **Date of Completion:**

Title: To implement a program in cpp for Max-Heap Tree.

Code:

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

class MaxHeap {
private:
    vector<int> heap;

    // Function to heapify up after insertion
    void heapifyUp(int index) {
        if (index == 0) return; // Base case
        int parent = (index - 1) / 2;
        if (heap[parent] < heap[index]) {
            swap(heap[parent], heap[index]);
            heapifyUp(parent); // Recursive call to ensure max-heap property
        }
    }

    // Function to heapify down after deletion
    void heapifyDown(int index) {
        int left = 2 * index + 1;
        int right = 2 * index + 2;
        int largest = index;

        if (left < heap.size() && heap[left] > heap[largest]) {
            largest = left;
        }
        if (right < heap.size() && heap[right] > heap[largest]) {
            largest = right;
        }
        if (largest != index) {
            swap(heap[index], heap[largest]);
            heapifyDown(largest); // Recursive call to maintain max-heap property
        }
    }
```

```
public:
    // Function to insert a new element into the heap
    void insert(int value) {
        heap.push_back(value);
        heapifyUp(heap.size() - 1); // Adjust position to maintain max-heap
    }

    // Function to remove and return the maximum element (root) from the heap
    int extractMax() {
        if (heap.empty()) {
            cout << "Heap is empty!" << endl;
            return -1;
        }
        int maxElement = heap[0];
        heap[0] = heap.back();
        heap.pop_back();
        heapifyDown(0); // Adjust position to maintain max-heap
        return maxElement;
    }

    // Function to display the elements of the heap
    void printHeap() {
        for (int i = 0; i < heap.size(); ++i) {
            cout << heap[i] << " ";
        }
        cout << endl;
    }
};

int main() {
    MaxHeap maxHeap;

    // Insert elements into the max heap
    maxHeap.insert(10);
    maxHeap.insert(20);
    maxHeap.insert(15);
    maxHeap.insert(30);
    maxHeap.insert(40);

    cout << "Max Heap after insertions: ";
    maxHeap.printHeap();

    // Extract maximum elements
```

```
 cout << "Extracted max: " << maxHeap.extractMax() << endl;
    cout << "Heap after extraction: ";
    maxHeap.printHeap();

    return 0;
}
```

Output:

Max Heap after insertions: 40 30 15 10 20

Extracted max: 40

Heap after extraction: 30 20 15 10

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R. Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 17

Date of Performance: **Date of Completion:**

Title: To implement a program in c for Min-Heap Tree.

Code:

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

// Function to heapify a subtree rooted with node i which is an index in arr[]
void minHeapify(vector<int>& arr, int n, int i) {
    int smallest = i; // Initialize smallest as root
    int left = 2 * i + 1; // left child
    int right = 2 * i + 2; // right child

    // If left child is smaller than root
    if (left < n && arr[left] < arr[smallest])
        smallest = left;

    // If right child is smaller than smallest so far
    if (right < n && arr[right] < arr[smallest])
        smallest = right;

    // If smallest is not root
    if (smallest != i) {
        swap(arr[i], arr[smallest]);

        // Recursively heapify the affected sub-tree
        minHeapify(arr, n, smallest);
    }
}

// Function to build a min heap from an array
void buildMinHeap(vector<int>& arr, int n) {
    // Start from the last non-leaf node and heapify all nodes in reverse order
    for (int i = n / 2 - 1; i >= 0; i--) {
        minHeapify(arr, n, i);
    }
```

```
}

// Function to print an array
void printArray(const vector<int>& arr) {
    for (int i = 0; i < arr.size(); ++i)
        cout << arr[i] << " ";
    cout << endl;
}

int main() {
    // Test with a random array
    vector<int> randomArray = {4, 10, 3, 5, 15};
    int n1 = randomArray.size();

    cout << "Original Random Array: ";
    printArray(randomArray);

    // Build heap
    buildMinHeap(randomArray, n1);

    cout << "Min Heap from Random Array: ";
    printArray(randomArray);
    cout << endl;

    // Test with a sorted array
    vector<int> sortedArray = {8, 6, 5, 4, 2};
    int n2 = sortedArray.size();

    cout << "Original Sorted Array: ";
    printArray(sortedArray);

    // Build heap
    buildMinHeap(sortedArray, n2);

    cout << "Min Heap from Sorted Array: ";
    printArray(sortedArray);

    return 0;
}
```

Output:

Original Random Array: 4 10 3 5 15
Min Heap from Random Array: 3 5 4 10 15

Original Sorted Array: 8 6 5 4 2
Min Heap from Sorted Array: 2 4 5 8 6

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R.Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 18

Date of Performance: **Date of Completion:**

Title: To implement a program in cpp for Depth First Traversal.

Code:

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

int main() {
   cout << "===== Program to demonstrate the DFS Traversal on a Graph, in CPP =====\n\n";

   // Variable declarations
   int cost[10][10] = {0}; // Adjacency matrix, initialized to 0
   int i, j, k, n, e, top = -1, v;
   int stk[10], visit[10] = {0}, visited[10] = {0};

   cout << "Enter the number of vertices in the Graph: ";
   cin >> n;

   cout << "\nEnter the number of edges in the Graph: ";
   cin >> e;

   cout << "\nEnter the start and end vertex of the edges:\n";
   for (k = 1; k <= e; k++) {
      cin >> i >> j;
      cost[i][j] = 1;
      cost[j][i] = 1; // For undirected graph
   }

   cout << "\nEnter the initial vertex to start the DFS traversal with: ";
   cin >> v;

   cout << "\nThe DFS traversal on the given graph is:\n";
   cout << v << " ";
   visited[v] = 1; // Mark the starting vertex as visited
```

```
k = 1;

 while (k < n) {
      for (j = n; j >= 1; j--) {
         if (cost[v][j] != 0 && visited[j] != 1 && visit[j] != 1) {
            visit[j] = 1;
            stk[++top] = j; // Push vertex onto the stack
         }
      }

      if (top == -1) {
         break; // If stack is empty, traversal is complete
      }

      v = stk[top--]; // Pop vertex from the stack
      cout << v << " ";
      visit[v] = 0;  // Mark it as removed from the visit stack
      visited[v] = 1; // Mark it as visited
      k++;
   }

cout << "\n";
   return 0;
}
```

Output:

```
===== Program to demonstrate the DFS Traversal on a Graph, in CPP =====

Enter the number of vertices in the Graph: 5

Enter the number of edges in the Graph: 4

Enter the start and end vertex of the edges:
1 2
1 3
2 4
3 5

Enter the initial vertex to start the DFS traversal with: 1

The DFS traversal on the given graph is:
1 3 5 2 4
```

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R.Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 19

Date of Performance: **Date of Completion:**

Title: To implement a program in cpp for Breadth First Traversal.

Code:

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

int n, i, j, visited[10], queue[10], front = -1, rear = -1;
int adj[10][10];

void bfs(int v) {
   // Mark the starting node as visited
   visited[v] = 1;
   queue[++rear] = v; // Enqueue the starting vertex

   while (front < rear) {
      // Dequeue a vertex and explore its adjacent nodes
      int current = queue[++front];

      // Explore all adjacent vertices of the current vertex
      for (i = 1; i <= n; i++) {
         if (adj[current][i] && !visited[i]) {
            queue[++rear] = i;  // Enqueue the unvisited adjacent vertex
            visited[i] = 1;      // Mark it as visited
         }
      }
   }
}

int main() {
   int v;
   cout << "Enter the number of vertices: ";
   cin >> n;

   // Initialize visited array and queue
   for (i = 1; i <= n; i++) {
      visited[i] = 0;
```

```
 queue[i] = 0;
   }

   cout << "Enter graph data in matrix form:\n";
   for (i = 1; i <= n; i++) {
      for (j = 1; j <= n; j++) {
         cin >> adj[i][j];
      }
   }

   cout << "Enter the starting vertex: ";
   cin >> v;

bfs(v);

   cout << "The nodes which are reachable are:\n";
   bool allVisited = true;
   for (i = 1; i <= n; i++) {
      if (visited[i]) {
         cout << i << "\t";
      } else {
         allVisited = false;
      }
   }

   if (!allVisited) {
      cout << "\nBFS is not possible. Not all nodes are reachable.\n";
   }

   return 0;
}
```

Output:

```
Enter the number of vertices: 5
Enter graph data in matrix form:
0 1 0 0 1
1 0 1 0 0
1 1 0 1 0
0 0 1 0 0
1 0 0 0 0
Enter the starting vertex: 1
The node which are reachable are:
1       2       3       4       5
```

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R. Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 20

Date of Performance: **Date of Completion:**

Title: To implement a program in cpp for obtaining shortest path using Dijkstra Algorithm.

Code:

```
#include <iostream>
#include <vector>
#include <set>
#include <climits>

using namespace std;

// Define infinity as a large value for initialization
#define INF INT_MAX

// Function to implement Dijkstra's algorithm
void dijkstra(int V, vector<pair<int, int>> adj[], int src) {
    // Create a distance array and initialize all distances to infinity
    vector<int> dist(V, INF);
    dist[src] = 0;

    // Use a set to keep track of vertices that are being processed
    set<pair<int, int>> s;
    s.insert({0, src}); // Insert source with distance 0

    while (!s.empty()) {
        // Get the vertex with the minimum distance
        int u = s.begin()->second;
        s.erase(s.begin());

        // Explore the neighbors of vertex u
        for (auto neighbor : adj[u]) {
            int v = neighbor.first;
            int weight = neighbor.second;

          // If a shorter path to v is found, update the distance
          if (dist[u] + weight < dist[v]) {
            // Remove the old pair and insert the new one with updated distance
```

```
            if (dist[v] != INF) {
                s.erase(s.find({dist[v], v}));
            }
            dist[v] = dist[u] + weight;
            s.insert({dist[v], v});
        }
    }
}

// Print the shortest distances from the source
cout << "Vertex\tDistance from Source\n";
for (int i = 0; i < V; i++) {
    if (dist[i] == INF)
        cout << i << "\tINF\n"; // If no path exists
    else
        cout << i << "\t" << dist[i] << endl;
}
}

int main() {
    // Number of vertices
    int V = 9;

    // Adjacency list representation of the graph
    vector<pair<int, int>> adj[V];

    // Add edges (undirected graph)
    adj[0].push_back({1, 6});
    adj[0].push_back({7, 8});
    adj[1].push_back({0, 6});
    adj[1].push_back({2, 8});
    adj[1].push_back({7, 13});
    adj[2].push_back({1, 8});
    adj[2].push_back({3, 7});
    adj[2].push_back({5, 6});
    adj[2].push_back({8, 2});
    adj[3].push_back({2, 7});
    adj[3].push_back({4, 9});
    adj[3].push_back({5, 14});
    adj[4].push_back({3, 9});
    adj[4].push_back({5, 10});
    adj[5].push_back({2, 6});
    adj[5].push_back({3, 14});
    adj[5].push_back({4, 10});
    adj[5].push_back({6, 2});
```

```
        adj[6].push_back({5, 2});
    adj[6].push_back({7, 1});
    adj[6].push_back({8, 6});
    adj[7].push_back({0, 8});
    adj[7].push_back({1, 13});
    adj[7].push_back({6, 1});
    adj[7].push_back({8, 7});
    adj[8].push_back({2, 2});
    adj[8].push_back({6, 6});
    adj[8].push_back({7, 7});

    // Run Dijkstra's algorithm from vertex 0
    dijkstra(V, adj, 0);

    return 0;
}
```

Output:

```
Vertex  Distance from Source
0       0
1       6
2       14
3       13
4       21
5       20
6       22
7       8
8       12
```

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R .Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 21

Date of Performance: **Date of Completion:**

Title: To implement a program in c for obtaining shortest Path using Floyd-Warshall Algorithm.

Code:

```
#include <iostream>
#include <iomanip> // for std::setw
#define V 4
#define INF 99999 // Define Infinite as a large enough value

using namespace std;

// A function to print the solution matrix
void printSolution(int dist[][V]);

// Solves the all-pairs shortest path problem using Floyd Warshall algorithm
void floydWarshall(int dist[][V]) {
   int i, j, k;

// Add all vertices one by one to the set of intermediate vertices
// Before start of an iteration, we have shortest distances between all pairs of vertices
 // such that the shortest distances consider only the vertices in set {0, 1, 2, .. k-1} as intermediate vertices
  // After the end of an iteration, vertex no. k is added to the set of intermediate vertices
 for (k = 0; k < V; k++) {
 // Pick all vertices as source one by one
for (i = 0; i < V; i++) {
   // Pick all vertices as destination for the above picked source
  for (j = 0; j < V; j++) {
   // If vertex k is on the shortest path from i to j, then update the value of dist[i][j]
  if (dist[i][k] + dist[k][j] < dist[i][j])
    dist[i][j] = dist[i][k] + dist[k][j];
          }
         }
          }
```

```
// Print the shortest distance matrix
printSolution(dist);
}

/* A utility function to print solution */
void printSolution(int dist[][V]) {
   cout << "The following matrix shows the shortest distances between every pair of
vertices\n";
    for (int i = 0; i < V; i++) {
      for (int j = 0; j < V; j++) {
 if (dist[i][j] == INF)
  cout << setw(7) << "INF"; // Formatting for better output
         else
 cout << setw(7) << dist[i][j];
 }
 cout << endl;
}
}

int main() {
  // Let us create the following weighted graph
  //     10
  // (0)------->(3)
 //  |         /|\
//  5 |       | |
 //  | |1     | |
          //  \|/       | |
 /        / (1)------->(2)
          //      3

           int graph[V][V] = { { 0, 5, INF, 10 },
            { INF, 0, 3, INF },
            { INF, INF, 0, 1 },
            { INF, INF, INF, 0 } };

          // Function call
           floydWarshall(graph);

          return 0;
          }
```

Output:

The following matrix shows the shortest distances between every pair of vertices

```
 0      5    INF     10
INF     0      3     INF
INF    INF     0      1
INF    INF    INF     0
```

Submitted By :

Sign :

Name :

Roll No :

Checked By :

Asst. Prof. Dhanshree R.Shinde

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 22

Date of Performance: **Date of Completion:**

Title: To implement a program in cpp for Minimum spanning tree using Kruskal Algorithm.

Code:

//Kruskal

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

// Edge structure to represent an edge
struct Edge {
    int u, v, weight;
};

// Disjoint Set Union (DSU) or Union-Find structure
class DSU {
public:
    vector<int> parent, rank;

    DSU(int n) {
        parent.resize(n);
        rank.resize(n, 0);
        for (int i = 0; i < n; i++) {
            parent[i] = i;
        }
    }

    // Find the parent of a node with path compression
    int find(int u) {
        if (u != parent[u]) {
            parent[u] = find(parent[u]);  // Path compression
        }
        return parent[u];
    }
```

```
    // Union by rank
    void unionSet(int u, int v) {
    int rootU = find(u);
     int rootV = find(v);

     if (rootU != rootV) {
      // Union by rank (attach the smaller tree under the larger one)
    if (rank[rootU] > rank[rootV]) {
      parent[rootV] = rootU;
     } else if (rank[rootU] < rank[rootV]) {
     parent[rootU] = rootV;
     } else {
parent[rootV] = rootU;
rank[rootU]++;
    }
    }
    }
    };

    // Function to implement Kruskal's algorithm
    int kruskal(vector<Edge>& edges, int n) {
        // Sort edges by their weights in non-decreasing order
        sort(edges.begin(), edges.end(), [](Edge a, Edge b) {
            return a.weight < b.weight;
        });

        DSU dsu(n);  // Initialize DSU for n vertices
        int mstWeight = 0;  // Total weight of the MST
        vector<Edge> mstEdges;  // To store the edges of the MST

        // Go through the sorted edges and add them to the MST if no cycle is formed
        for (Edge& edge : edges) {
            int u = edge.u;
            int v = edge.v;
            int weight = edge.weight;

            // If u and v are in different sets, add this edge to the MST
            if (dsu.find(u) != dsu.find(v)) {
                dsu.unionSet(u, v);
                mstEdges.push_back(edge);
                mstWeight += weight;
            }
                }
```

```
// Print the MST edges and total weight
    cout << "\nEdges in the Minimum Spanning Tree (MST):\n";
    for (Edge& edge : mstEdges) {
        cout << edge.u << " - " << edge.v << " : " << edge.weight << endl;
    }

    return mstWeight;
}

int main() {
    int n, e;

    // Input the number of vertices and edges
    cout << "Enter the number of vertices: ";
    cin >> n;
    cout << "Enter the number of edges: ";
    cin >> e;

    vector<Edge> edges(e);

    // Input the edges (u, v, weight)
    cout << "Enter the edges (u, v, weight):\n";
    for (int i = 0; i < e; i++) {
        cin >> edges[i].u >> edges[i].v >> edges[i].weight;
    }

    // Call Kruskal's algorithm to find the MST
    int mstWeight = kruskal(edges, n);

    // Print the total weight of the MST
    cout << "\nTotal weight of the Minimum Spanning Tree: " << mstWeight
<< endl;

    return 0;
}
```

Output:

```
Enter the number of vertices: 4
Enter the number of edges: 5
Enter the edges (u, v, weight):
0 1 10
0 2 6
0 3 5
1 3 15
2 3 4
Edges in the Minimum Spanning Tree (MST):
3 - 2 : 4
0 - 3 : 5
0 - 1 : 10

Total weight of the Minimum Spanning Tree: 19
```

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R .Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 23

Date of Performance: **Date of Completion:**

Title:To implement a program in cpp for Minimum spanning tree using Prims Algorithm.

Code:

```
#include <iostream>
#include <climits> // for INT_MAX

using namespace std;

int main() {
int a, b, n, ne = 1, i, j, min, cost[10][10], mincost = 0;

// Taking the number of vertices as input
cout << "\nEnter the number of vertices: ";
cin >> n;

cout << "\nEnter the adjacency matrix (use 0 for no edge): \n";

 // Input the adjacency matrix
 for (i = 1; i <= n; i++) {
for (j = 1; j <= n; j++) {
 cin >> cost[i][j];
if (cost[i][j] == 0) {
  cost[i][j] = INT_MAX; // Replace 0 with a large number (infinity)
 }
 }
 }

        // Prim's Algorithm to find the MST
while (ne < n) {
min = INT_MAX;

// Search for the minimum edge
for (i = 1; i <= n; i++) {
        for (j = 1; j <= n; j++) {
        if (cost[i][j] < min) {
        min = cost[i][j];
```

```
 a = i;
 b = j;
}
}
}

// Print the edge that is part of MST
cout << "Edge (" << a << ", " << b << ") = " << min << endl;

 // Add the minimum weight edge to the total cost
 mincost += min;

// Mark the edge as processed by setting the cost to a large number (effectively
removing it)
cost[a][b] = cost[b][a] = INT_MAX;

// Increment the number of edges in the MST
  ne++;
  }

  // Output the total minimum cost of the MST
 cout << "\nMinimum Spanning Tree total weight = " << mincost << endl;

  return 0;
}
```

Output:

Enter the number of vertices: 4

Enter the adjacency matrix (use 0 for no edge):
0 10 999 30
10 0 50 999
999 50 0 20
30 999 20 0
Edge (1, 2) = 10
Edge (1, 4) = 30
Edge (3, 4) = 20

Minimum Spanning Tree total weight = 60

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R.Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 24

Date of Performance: **Date of Completion:**

Title:To implement a program in cpp for Hash Table.

Code:

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

// HashTable class
class HashTable {
private:
   static const int hashGroups = 10; // Number of groups (buckets)
   list<int> table[hashGroups];      // Array of lists to handle collisions

   // Hash function to calculate hash value
   int hashFunction(int key) {
      return key % hashGroups;
   }

public:
   // Insert a key into the hash table
   void insert(int key) {
      int index = hashFunction(key);
      table[index].push_back(key);
   }

   // Delete a key from the hash table
   void remove(int key) {
      int index = hashFunction(key);
      table[index].remove(key);
   }

   // Search for a key in the hash table
   bool search(int key) {
      int index = hashFunction(key);
      for (auto it : table[index]) {
         if (it == key)
            return true;
      }
```

```
 return false;
 }

 // Display the hash table
 void display() {
     for (int i = 0; i < hashGroups; i++) {
cout << "Bucket " << i << ": ";
for (auto it : table[i]) {
cout << it << " -> ";
}
Cout << "null" << endl;
}
 }
};

int main() {
 HashTable ht;

// Insert elements
 ht.insert(10);
ht.insert(20);
ht.insert(15);
 ht.insert(7);
ht.insert(25);

 // Display hash table
cout << "Hash Table: " << endl;
ht.display();

    // Search for a key
    int key = 15;
    cout << "\nSearching for key " << key << ": ";
    if (ht.search(key)) {
        cout << "Found" << endl;
    } else {
        cout << "Not Found" << endl;
    }

    // Remove a key
    key = 20;
    cout << "\nRemoving key " << key << endl;
    ht.remove(key);

          // Display updated hash table
```

```
 cout << "\nUpdated Hash Table: " << endl;

ht.display();

 return 0;
}
```

Output:

```
Hash Table:
Bucket 0: null
Bucket 1: null
Bucket 2: null
Bucket 3: null
Bucket 4: null
Bucket 5: null
Bucket 6: null
Bucket 7: 7 -> null
Bucket 8: null
Bucket 9: 10 -> 20 -> 15 -> 25 -> null

Searching for key 15: Found

Removing key 20

Updated Hash Table:
Bucket 0: null
Bucket 1: null
Bucket 2: null
Bucket 3: null
Bucket 4: null
Bucket 5: null
Bucket 6: null
Bucket 7: 7 -> null
Bucket 8: null
Bucket 9: 10 -> 15 -> 25 -> null
```

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R .Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 25

Date of Performance: **Date of Completion:**

Title:To implement a program in cpp for Linear Search using array.

Code:

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

// Function to perform linear search
int linearSearch(int arr[], int n, int target) {
   for (int i = 0; i < n; ++i) {
      if (arr[i] == target)
         return i; // Return the index if target is found
   }
   return -1; // Return -1 if target is not found
}

int main() {
   int arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91};
   int n =

sizeof(arr) / sizeof(arr[0]);
   int target = 23;

   int result = linearSearch(arr, n, target);
   if (result != -1)
      cout << "Element found at index " << result << endl;
   else
      cout << "Element not found in the array" << endl;

   return 0;
}
```

Output:

```
How many elements u want to enter:
5
Enter array elements:
10
20
30
40
50
Enter element to search: 30
Element found at index 2
```

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R.Shinde**

Roll No :

SBT's College of Engineering & Technology, Bambhori, Jalgaon
Department of Computer Applications
Practical: 26

Date of Performance: **Date of Completion:**

Title:To implement a program in cpp for Binary Search using array.

Code:

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

// Function to perform binary search
int binarySearch(int arr[], int left, int right, int target) {
    while (left <= right) {
        int mid = left + (right - left) / 2;

        // Check if target is present at mid
        if (arr[mid] == target)
            return mid;

        // If target is greater, ignore left half
        if (arr[mid] < target)
            left = mid + 1;
        else
            right = mid - 1;
    }

    return -1; // Target is not present
}

int main() {
    int arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91};
    int n = sizeof(arr) / sizeof(arr[0]);
    int target = 23;

    int result = binarySearch(arr, 0, n - 1, target);
    if (result != -1)
        cout << "Element found at index " << result << endl;
    else
        cout << "Element not found in the array" << endl;

    return 0;
}
```

Output:

```
Enter number of elements:
5
Enter 5 integers:
10
20
30
40
50
Enter value to find: 50
50 found at location 5
```

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R.Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 27

Date of Performance: **Date of Completion:**

Title:To implement a program in cpp for Bubble Sort.

Code:

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

int main() {
   int array[100], n, i, j, swap;

   cout << "Enter number of elements: ";
   cin >> n;

   cout << "\nEnter " << n << " Numbers:\n";
   for (i = 0; i < n; i++)
      cin >> array[i];

   // Bubble Sort
   for (i = 0; i < n - 1; i++) {
      for (j = 0; j < n - i - 1; j++) {
         if (array[j] > array[j + 1]) {
            swap = array[j];
            array[j] = array[j + 1];
            array[j + 1] = swap;
         }
      }
   }

   cout << "\nSorted Array:\n";
   for (i = 0; i < n; i++)
      cout << array[i] << endl;

   return 0;
}
```

Output:

Enter number of elements: 5

Enter 5 Numbers:
42
25
33
17
8

Sorted Array:
8
17
25
33
42

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R. Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 28

Date of Performance: **Date of Completion:**

Title:To implement a program in cpp for Selection Sort.

Code:

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

/*
 * Function to swap two variables
 */
void swap(int *a, int *b) {
   int temp = *a;
   *a = *b;
   *b = temp;
}

/*
 * Selection sort function
 */
void selectionSort(int arr[], int size) {
   int i, j;
   for (i = 0; i < size; i++) {
      for (j = i; j < size; j++) {
         if (arr[i] > arr[j])
            swap(&arr[i], &arr[j]);
      }
   }
}

/*
 * Main Function
 */
int main() {
int array[10], size;

   cout << "How many numbers you want to sort: ";
   cin >> size;
```

```
 cout << "\nEnter " << size << " numbers:\n";

for (int i = 0; i < size; i++)
 cin >> array[i];

 selectionSort(array, size);

cout << "\nSorted array is:\n";
 for (int i = 0; i < size; i++)
cout << array[i] << endl;

 return 0;
}
```

Output:

How many numbers you want to sort: 5

Enter 5 numbers:
42
17
33
8
25
Sorted array is:
8
17
25
33
42

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R. Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 29

Date of Performance: **Date of Completion:**

Title: To implement a program in cpp for Insertion Sort.

Code:

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

int main() {
   int n, i, j, temp;
   int arr[64];

   cout << "Enter number of elements: ";
   cin >> n;

   cout << "\nEnter " << n << " integers:\n";
   for (i = 0; i < n; i++) {
      cin >> arr[i];
   }

   // Insertion Sort Algorithm
   for (i = 1; i < n; i++) {
      j = i;
      while (j > 0 && arr[j - 1] > arr[j]) {
         temp = arr[j];
         arr[j] = arr[j - 1];
         arr[j - 1] = temp;
         j--;
      }
   }

   cout << "\nSorted list in ascending order:\n";
   for (i = 0; i < n; i++) {
      cout << arr[i] << endl;
   }

   return 0;
}
```

Output:

Enter number of elements: 5

Enter 5 integers:
42
17
33
8
25
Sorted list in ascending order:
8
17
25
33
42

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R .Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 30

Date of Performance: **Date of Completion:**

Title: To implement a program in cpp for Radix Sort.

Code:

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

vector<int> radixSort(vector<int> arr, int size);
vector<int> countSort(vector<int> arr, int size, int exponent);
int maximum(const vector<int>& arr);
int minimum(const vector<int>& arr);

int main() {
    int size;
    cout << "Enter the array size: ";
    cin >> size;

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

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

arr = radixSort(arr, size);

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

 return 0;
```

```
}

vector<int> radixSort(vector<int> arr, int size) {
   int maxOfArr = maximum(arr);
   int exponent = 1;

while (exponent <= maxOfArr) {
arr = countSort(arr, size, exponent);
exponent *= 10;
}

 return arr;
}

vector<int> countSort(vector<int> arr, int size, int exponent) {
int range = 10; // For decimal numbers (0-9)
   vector<int> frequencyArray(range, 0);
   vector<int> newArr(size);

   // Count occurrences of digits at the current place
   for (int i = 0; i < size; i++) {
      frequencyArray[(arr[i] / exponent) % 10]++;
   }

   // Update frequency array to store cumulative counts
   for (int i = 1; i < range; i++) {
      frequencyArray[i] += frequencyArray[i - 1];
   }

// Build the sorted array
for (int i = size - 1; i >= 0; i--) {
  int pos = frequencyArray[(arr[i] / exponent) % 10] - 1;
newArr[pos] = arr[i];
frequencyArray[(arr[i] / exponent) % 10]--;
}

return newArr;
        }

int maximum(const vector<int>& arr) {
 int max = INT_MIN;
for (int num : arr) {
if (num > max) {
max = num;
```

```
}
}
return max;
}

int minimum(const vector<int>& arr) {
int min = INT_MAX;
for (int num : arr) {
if (num < min) {
min = num;
}
}
return min;
}
```

Output:

Enter the array size: 6

Enter the array elements:

arr[0] = 170

arr[1] = 45

arr[2] = 75

arr[3] = 90

arr[4] = 802

arr[5] = 24

Array before sorting:
arr[0] = 170
arr[1] = 45
arr[2] = 75
arr[3] = 90
arr[4] = 802
arr[5] = 24

Array after sorting:
arr[0] = 24
arr[1] = 45
arr[2] = 75
arr[3] = 90
arr[4] = 170
arr[5] = 802

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R .Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 31

Date of Performance: **Date of Completion:**

Title:To implement a program in cpp for Quick Sort.

Code:

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

// QuickSort Function
void quickSort(int *arr, int low, int high) {
    int i = low, j = high;
    int pivot = arr[(low + high) / 2]; // Choose middle element as pivot

    while (i <= j) {
        while (arr[i] < pivot) i++;
        while (arr[j] > pivot) j--;

        if (i <= j) {
            swap(arr[i], arr[j]); // Swap elements
            i++;
            j--;
        }
    }

    // Recursively sort the two halves
    if (low < j)
        quickSort(arr, low, j);
    if (i < high)
        quickSort(arr, i, high);
}

int main() {
    int n;

    cout << "Enter the number of elements in the array: ";
    cin >> n;

    int arr[n];
    cout << "Enter the elements of the array:\n";
    for (int i = 0; i < n; i++) {
        cout << "arr[" << i << "]: ";
```

```
        cin >> arr[i];
    }

    // Perform QuickSort
    quickSort(arr, 0, n - 1);

    // Output the sorted array
    cout << "The sorted array is:\n";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << endl;
    }

    return 0;
}
```

Output:

Enter the number of elements in the array: 6

Enter the elements of the array:
arr[0]: 42
arr[1]: 17
arr[2]: 33
arr[3]: 8
arr[4]: 25
arr[5]: 90

The sorted array is:
8
17
25
33
42
90

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R.Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 32

Date of Performance: **Date of Completion:**

Title:To implement a program in cpp for Merge Sort.

Code:

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

// Merge function to merge two halves
void merge(vector<int>& arr, int left, int mid, int right) {
    int size1 = mid - left + 1;
    int size2 = right - mid;

    // Create temporary arrays
    vector<int> Left(size1), Right(size2);

    // Copy data to temp arrays
    for (int i = 0; i < size1; i++) {
        Left[i] = arr[left + i];
    }
    for (int j = 0; j < size2; j++) {
        Right[j] = arr[mid + 1 + j];
    }

    // Merge the temp arrays back into arr
    int i = 0, j = 0, k = left;
    while (i < size1 && j < size2) {
        if (Left[i] <= Right[j]) {
            arr[k] = Left[i];
            i++;
        } else {
            arr[k] = Right[j];
            j++;
        }
        k++;
    }

    // Copy remaining elements of Left[] if any
```

```
while (i < size1) {
        arr[k] = Left[i];
        i++;
        k++;

  }

    // Copy remaining elements of Right[] if any
    while (j < size2) {
        arr[k] = Right[j];
        j++;
        k++;
    }
}


// Merge Sort function to divide and conquer

void mergeSort(vector<int>& arr, int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        // Recursively divide the array into two halves
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);

        // Merge the two halves
        merge(arr, left, mid, right);
    }
}

int main() {
int size;

 cout << "Enter the size: ";
    cin >> size;

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

    // Call Merge Sort
```

```
mergeSort(arr, 0, size - 1);

    // Print the sorted array
    cout << "The sorted array is:\n";



for (int i = 0; i < size; i++) {

cout << arr[i] << endl;
    }

    return 0;
}
```

Output:

```
Enter the size: 6
Enter the elements of the array:
34
7
23
90
12
5

The sorted array is:
5
7
12
23
34
90
```

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R.Shinde**

Roll No :

SSBT's College of Engineering & Technology, Bambhori, Jalgaon

Department of Computer Applications

Practical: 33

Date of Performance: **Date of Completion:**

Title:To implement a program in cpp for Heap Sort.

Code:

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

// Function prototypes
void heapify(int*, int, int);
void heapsort(int*, int);
void print_array(int*, int);

int main()
{
   int arr[] = {10, 3, 5, 16, 92, 12, 56, 43};
   int n = sizeof(arr) / sizeof(arr[0]);

cout << "\nArray before sorting:\n";
print_array(arr, n);

heapsort(arr, n);

cout << "\n\nArray after sorting:\n";
 print_array(arr, n);

return 0;
}

/* Function to perform heapify on the subtree with root at index i */
void heapify(int* arr, int n, int i)
{
   int largest = i;
   int left = 2 * i + 1;
   int right = 2 * i + 2;

   // Check if the left child is larger than the root
   if (left < n && arr[left] > arr[largest]) {
      largest = left;
   }
```

```
    // Check if the right child is larger than the largest so far
    if (right < n && arr[right] > arr[largest]) {
        largest = right;
    }

    // If the largest is not the root, swap them and continue heapifying
    if (largest != i) {
        swap(arr[i], arr[largest]);
        heapify(arr, n, largest);
    }
}

/* Function to perform heapsort */
void heapsort(int* arr, int n)
{
    // Build the max heap
    for (int i = n / 2 - 1; i >= 0; i--) {
        heapify(arr, n, i);
    }

    // One by one extract elements from the heap
    for (int i = n - 1; i >= 0; i--) {
        // Move the current root (largest) to the end
        swap(arr[0], arr[i]);

        // Call heapify on the reduced heap
        heapify(arr, i, 0);
    }
}

/* Function to print the array */
void print_array(int* arr, int n)
{
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}
```

Output:

int arr[] = {10, 3, 5, 16, 92, 12, 56, 43};

Array before sorting:
10 3 5 16 92 12 56 43

Array after sorting:
3 5 10 12 16 43 56 92

Submitted By : **Checked By :**

Sign :

Name : **Asst. Prof. Dhanshree R. Shinde.**

Roll No :