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

Department of Computer Applications

Practical: 01

DOP: **DOC:**

Title: Program for finding summation of array element

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

int main()
{
   int A[100]; // Declare Array
   int n;
   int sum = 0;

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

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

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

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

    return 0;
}
```

Output

```
Enter size of array: 5
Enter array element: 2
Enter array element: 4
Enter array element: 6
Enter array element: 8
Enter array element: 10
Summation of Array is = 30
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 02

DOP: **DOC:**

Title: Program to read a matrix and compute the addition of Matrix

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

int main()
{
   int i, j, frst[3][3], second[3][3], sum[3][3];

   cout << "Enter the elements of first matrix" << endl;
   for (i = 0; i < 3; i++)
   {
      for (j = 0; j < 3; j++)
      {
         cin >> frst[i][j];
      }
   }

   cout << "Enter the elements of second matrix" << endl;
   for (i = 0; i < 3; i++)
   {
      for (j = 0; j < 3; j++)
      {
         cin >> second[i][j];
      }
   }

   cout << "Sum of entered matrices:-" << endl;
   for (i = 0; i < 3; i++)
   {
      for (j = 0; j < 3; j++)
      {
         sum[i][j] = frst[i][j] + second[i][j];
         cout << sum[i][j] << "\t";
      }
      cout << endl;
   }
```

```
    return 0;
}
```

Output

```
Enter the elements of first matrix
1 2 3
4 5 6
7 8 9
Enter the elements of second matrix
9 8 7
6 5 4
3 2 1

Sum of entered matrices:-
10 10 10
10 10 10
10 10 10
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 03

DOP: **DOC:**

Title: Program For Stack

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

int stack[MAXSIZE];
int top = -1; // index pointing to the top of stack

void push(int y)
{
   if (top + 1 == MAXSIZE)
   {
      cout << "\nNo Space Stack Is Full\n";
      return;
   }
   else
   {
      top++;
      stack[top] = y;
      cout << "\n" << y << " added to Stack\n";
   }
}

void pop()
{
   int a;
   if (top == -1)
   {
      cout << "\nStack Is Empty\n";
   }
   else
   {
      a = stack[top];
      top--;
      cout << "\nValue returned from stack: " << a << endl;
   }
}

void display()
{
   int i;
```

```
    if (top == -1)
    {
        cout << "\nStack Is Empty\n";
    }
    else
    {
        cout << "\n\n\nContents of the stack are:\n";
        for (i = top; i >= 0; i--)
        {
            cout << stack[i] << endl;
        }
    }
}

int main()
{
    int i, num, menuSelect;

    cout << "\n\t\tProgram for Implementation of Stack using array\n";
    do
    {
        cout << "\n\n\tMain Menu:\n1. Add element to stack (Push)\n2. Delete element from the
stack (Pop)\n3. Display Stack\n4. Exit";
        cout << "\nSelect menu: ";
        cin >> menuSelect;

        switch (menuSelect)
        {
            case 1:
                cout << "\nEnter a number: ";
                cin >> num;
                push(num);
                break;
            case 2:
                pop();
                break;
            case 3:
                display();
                break;
            case 4:
                cout << "\nExiting the program" << endl;
                break;
            default:
                cout << "Invalid menu item selected.";
        }
    } while (menuSelect != 4);

    return 0;
}
```

Output

Program for Implementation of Stack using array

Main Menu:
1. Add element to stack (Push)
2. Delete element from the stack (Pop)
3. Display Stack
4. Exit
Select menu: 1

Enter a number: 10

10 added to Stack

Main Menu:
1. Add element to stack (Push)
2. Delete element from the stack (Pop)
3. Display Stack
4. Exit
Select menu: 1

Enter a number: 20

20 added to Stack

Main Menu:
1. Add element to stack (Push)
2. Delete element from the stack (Pop)
3. Display Stack
4. Exit
Select menu: 3

Contents of the stack are:
20
10

Main Menu:
1. Add element to stack (Push)
2. Delete element from the stack (Pop)
3. Display Stack
4. Exit
Select menu: 2

Value returned from stack: 20

Main Menu:
1. Add element to stack (Push)
2. Delete element from the stack (Pop)
3. Display Stack
4. Exit
Select menu: 3

Contents of the stack are:
10

Main Menu:
1. Add element to stack (Push)
2. Delete element from the stack (Pop)
3. Display Stack
4. Exit
Select menu: 4

Exiting the program

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 04

DOP: **DOC:**

Title: Program For postfx to infx conversion

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

int main()
{
   char stk[50][50], p[30], temp[50], opr[10], op1[10], op2[10];
   int top = -1, i = 0;

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

   while (p[i] != '\0')
   {
      if (p[i] == '+' || p[i] == '-' || p[i] == '*' || p[i] == '/' || p[i] == '^')
      {
         strcpy(op2, stk[top--]);
         strcpy(op1, stk[top--]);
         opr[0] = p[i];
         opr[1] = '\0';
         strcpy(temp, "(");
         strcat(temp, op1);
         strcat(temp, opr);
         strcat(temp, op2);
         strcat(temp, ")");
         strcpy(stk[++top], temp);
      }
      else
      {
         temp[0] = p[i];
         temp[1] = '\0';
         strcpy(stk[++top], temp);
      }
      i++;
   }

   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 :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 05

DOP: **DOC:**

Title: Progrm for Infx to Postfx Conversion

```
#include <iostream>
#include <cstdlib>
#include <cmath>
#include <cstring>
#define SIZE 100
using namespace std;

char stack[SIZE] = "";
int top = -1;

char pop(void)
{
   char a;
   a = stack[top];
   top--;
   return a;
}

void push(char y)
{
   top++;
   stack[top] = y;
}

int prcd(char a, char b)
{
   if (prior(a) > prior(b))
      return 1;
   else
      return 0;
}

int isoperand(char c)
{
   if ((c != '+' && c != '-' && c != '*' && c != '/' && c != '(' && c != ')' && c != '^'))
```

```
        return 1;
    else
        return 0;
}

int prior(char i)
{
    switch (i)
    {
    case '+':
        return 1;
    case '-':
        return 1;
    case '*':
        return 2;
    case '/':
        return 2;
    case '^':
        return 3;
    case '#':
        return -1;
    }
    return 0;
}

int main()
{
    char infx[100], c, postfx[100] = "", s;
    int i = 0, k = 0;
    push('#');

    cout << "\nEnter the infix expression: ";
    cin >> infx;

    while (infx[i] != '\0')
    {
        c = infx[i++];

        if (isoperand(c))
            postfx[k++] = c;
        else
        {
            if (c == '(')
            {
                push(c);
                continue;
```

```
            }
            if (c == ')')
            {
               while ((s = pop()) != '(')
                  postfx[k++] = s;
               continue;
            }
            if (prcd(c, stack[top]))
               push(c);
            else
            {
               while (!prcd(c, stack[top]))
                  postfx[k++] = pop();
               push(c);
            }
         }
      }

      while (top != 0)
         postfx[k++] = pop();
      postfx[k] = '\0';

      cout << "\nThe postfix string is: " << postfx << endl;

      return 0;
}
```

Output

```
Enter the infix expression (A+B)*C
the postfix string is AB+C*
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 06

DOP: **DOC:**

Title: Program For Queue Implementation through Array

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

int queue[MAXSIZE];
int front = 0;
int rear = 0;

int isFull()
{
   if (rear == MAXSIZE)
      return 1;
   else
      return 0;
}

int isEmpty()
{
   if (front == rear)
      return 1;
   else
      return 0;
}

void addq(int y)
{
   if (isFull() == 1)
   {
      cout << "\nQueue Full\n";
      return;
   }
   else
   {
      queue[rear] = y;
      rear++;
      cout << "\n" << y << " added to Queue\n";
   }
```

```
}

void delq()
{
   int a;
   if (isEmpty() == 1)
   {
      cout << "\nQueue Empty\n";
   }
   else
   {
      a = queue[front];
      front++;
      cout << "\nValue returned from queue: " << a << endl;
   }
}

void display()
{
   int i;
   if (isEmpty() == 1)
   {
      cout << "\nQueue Empty\n";
   }
   else
   {
      cout << "\n\n\nContents of the queue are:\n";
      for (i = front; i < rear; i++)
      {
         cout << queue[i];
         if (front == i)
            cout << "\t<--- Front" << endl;
         else
         {
            if ((rear - 1) == i)
               cout << "\t<--- Rear" << endl;
            else
               cout << endl;
         }
      }
   }
}

int main()
{
   int i, num, menuSelect;
```

```
   cout << "\n\t\tProgram for queue using array\n";
   do
   {
      cout << "\n\n\tMain Menu:\n1. Add element\n2. Delete element\n3. Display Queue\n4.
Exit";
      cout << "\nSelect menu: ";
      cin >> menuSelect;

      switch (menuSelect)
      {
         case 1:
            cout << "\nEnter a number: ";
            cin >> num;
            addq(num);
            break;
         case 2:
            delq();
            break;
         case 3:
            display();
            break;
         case 4:
            cout << "\nExiting the program" << endl;
            break;
         default:
            cout << "Invalid menu item selected.";
      }
   } while (menuSelect != 4);

   return 0;
}
```

Output

```
      Program for queue using array


      Main Menu:
1. Add element
2. Delete element
3. Display Queue
4. Exit
Select menu: 1

Enter a number: 5
```

5 added to Queue

Main Menu:
1. Add element
2. Delete element
3. Display Queue
4. Exit
Select menu: 1

Enter a number: 3

3 added to Queue

Main Menu:
1. Add element
2. Delete element
3. Display Queue
4. Exit
Select menu: 3

Contents of the queue are:
5 <--- Front
3 <--- Rear

Main Menu:
1. Add element
2. Delete element
3. Display Queue
4. Exit
Select menu: 2

Value returned from queue: 5

Main Menu:
1. Add element
2. Delete element
3. Display Queue
4. Exit
Select menu: 3

Contents of the queue are:
3 <--- Front
<--- Rear

Main Menu:
1. Add element
2. Delete element
3. Display Queue
4. Exit
Select menu: 2

Value returned from queue: 3

Main Menu:
1. Add element
2. Delete element
3. Display Queue
4. Exit
Select menu: 2

Queue Empty

Main Menu:
1. Add element
2. Delete element
3. Display Queue
4. Exit
Select menu: 4

Exiting the program

6)

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 07

DOP: **DOC:**

Title: Program For singly lined list

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

struct Node
{
   int data;
   Node* next;
};

Node* head = NULL;

void begInsert()
{
   Node* ptr = new Node();
   int item;

   if (ptr == NULL)
   {
      cout << "\nOVERFLOW\n";
      return;
   }

   cout << "\nEnter value: ";
   cin >> item;

   ptr->data = item;
   ptr->next = head;
   head = ptr;

   cout << "\nNode inserted\n";
}

void lastInsert()
{
   Node* ptr = new Node();
   int item;
```

```
    if (ptr == NULL)
    {
        cout << "\nOVERFLOW\n";
        return;
    }

    cout << "\nEnter value: ";
    cin >> item;

    ptr->data = item;
    ptr->next = NULL;

    if (head == NULL)
    {
        head = ptr;
        cout << "\nNode inserted\n";
        return;
    }

    Node* temp = head;
    while (temp->next != NULL)
    {
        temp = temp->next;
    }

    temp->next = ptr;
    cout << "\nNode inserted\n";
}

void randomInsert()
{
    Node* ptr = new Node();
    int item, loc, i;

    if (ptr == NULL)
    {
        cout << "\nOVERFLOW\n";
        return;
    }

    cout << "\nEnter value: ";
    cin >> item;

    cout << "\nEnter the location after which you want to insert: ";
    cin >> loc;
```

```
    ptr->data = item;

    Node* temp = head;
    for (i = 0; i < loc; i++)
    {
        temp = temp->next;
        if (temp == NULL)
        {
            cout << "\nCan't insert\n";
            return;
        }
    }

    ptr->next = temp->next;
    temp->next = ptr;

    cout << "\nNode inserted\n";
}

void beginDelete()
{
    if (head == NULL)
    {
        cout << "\nList is empty\n";
        return;
    }

    Node* ptr = head;
    head = ptr->next;
    delete ptr;

    cout << "\nNode deleted from the beginning\n";
}

void lastDelete()
{
    if (head == NULL)
    {
        cout << "\nList is empty\n";
        return;
    }

    Node* ptr = head;
    Node* prev = NULL;

    if (ptr->next == NULL)
```

```
    {
        head = NULL;
        delete ptr;
        cout << "\nOnly node of the list deleted\n";
        return;
    }

    while (ptr->next != NULL)
    {
        prev = ptr;
        ptr = ptr->next;
    }

    prev->next = NULL;
    delete ptr;

    cout << "\nDeleted Node from the last\n";
}

void randomDelete()
{
    int loc, i;
    cout << "\nEnter the location of the node after which you want to perform deletion: ";
    cin >> loc;

    Node* ptr = head;
    Node* prev = NULL;

    for (i = 0; i < loc; i++)
    {
        prev = ptr;
        ptr = ptr->next;

        if (ptr == NULL)
        {
            cout << "\nCan't delete\n";
            return;
        }
    }

    prev->next = ptr->next;
    delete ptr;

    cout << "\nDeleted node " << loc + 1 << endl;
}
```

```
void search()
{
   Node* ptr = head;
   int item, i = 0, flag = 0;

   if (ptr == NULL)
   {
      cout << "\nEmpty List\n";
      return;
   }

   cout << "\nEnter item which you want to search: ";
   cin >> item;

   while (ptr != NULL)
   {
      if (ptr->data == item)
      {
         cout << "Item found at location " << i + 1 << endl;
         flag = 1;
      }
      i++;
      ptr = ptr->next;
   }

   if (flag == 0)
   {
      cout << "Item not found\n";
   }
}

void display()
{
   Node* ptr = head;

   if (ptr == NULL)
   {
      cout << "Nothing to print\n";
      return;
   }

   cout << "\nPrinting values:\n";
   while (ptr != NULL)
   {
      cout << ptr->data << endl;
      ptr = ptr->next;
```

```
    }
}
```

Output

```
********Main Menu********
Choose one option from the following list ...
==============================================
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specifed location
7.Search for an element
8.Show
9.Exit
Enter your choice?
1
Enter value
1
Node inserted
********Main Menu********
Choose one option from the following list ...
==============================================
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specifed location
7.Search for an element
8.Show
9.Exit
Enter your choice?
2
Enter value?
2
Node inserted
********Main Menu********

Choose one option from the following list ...
==============================================
1.Insert in begining
2.Insert at last
3.Insert at any random location
```

4.Delete from Beginning
5.Delete from last
6.Delete node after specifed location
7.Search for an element
8.Show
9.Exit
Enter your choice?
3
Enter element value1
Enter the location after which you want to insert 1
Node inserted
********Main Menu********
Choose one option from the following list ...
==========
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specifed location
7.Search for an element
8.Show
9.Exit
Enter your choice?
8
printing values . . . . .
1
2
1
********Main Menu********
Choose one option from the following list ...
==========
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specifed location
7.Search for an element
8.Show
9.Exit
Enter your choice?
2
Enter value?
123
Node inserted

********Main Menu********
Choose one option from the following list ...
==========
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specifed location
7.Search for an element
8.Show
9.Exit
Enter your choice?
1
Enter value
1234
Node inserted
********Main Menu********
Choose one option from the following list ...
==========
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specifed location
7.Search for an element
8.Show
9.Exit
Enter your choice?
4
Node deleted from the begining ...
********Main Menu********
Choose one option from the following list ...
==========
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specifed location
7.Search for an element
8.Show
9.Exit
Enter your choice?
5

```
Deleted Node from the last ...
********Main Menu********
Choose one option from the following list ...
==============================================
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specifed location
7.Search for an element
8.Show
9.Exit
Enter your choice?
6
Enter the location of the node after which you want to perform deletion
1
Deleted node 2
********Main Menu********
Choose one option from the following list ...
==============================================
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specifed location
7.Search for an element
8.Show
9.Exit
Enter your choice?
8
printing values . . . . .
1
1
********Main Menu********
Choose one option from the following list ...
==============================================
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specifed location
7.Search for an element
8.Show
```

9.Exit
Enter your choice?
7
Enter item which you want to search?
1
item found at location 1
item found at location 2
********Main Menu********
Choose one option from the following list ...
==========
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specifed location
7.Search for an element
8.Show
9.Exit
Enter your choice?
9
*/

7)

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 08

DOP: **DOC:**

Title: Program For Dijkstra Shortest path algo

```
#include <iostream>
#include <vector>
#include <queue>
#include <limits>

using namespace std;

#define INF numeric_limits<int>::max()

// Graph edge representation
struct Edge {
    int destination;
    int weight;
};

// Dijkstra's algorithm
void dijkstra(const vector<vector<Edge>>& graph, int source) {
    int numVertices = graph.size();
    vector<int> distances(numVertices, INF);
    vector<bool> visited(numVertices, false);
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;

    distances[source] = 0;
    pq.push(make_pair(0, source));

    while (!pq.empty()) {
        int u = pq.top().second;
        pq.pop();

        if (visited[u])
            continue;

        visited[u] = true;

        for (const auto& edge : graph[u]) {
            int v = edge.destination;
```

```
            int weight = edge.weight;

            if (distances[u] + weight < distances[v]) {
                distances[v] = distances[u] + weight;
                pq.push(make_pair(distances[v], v));
            }
        }
    }

    // Print shortest distances
    cout << "Vertex\tDistance from Source\n";
    for (int i = 0; i < numVertices; ++i) {
        cout << i << "\t" << distances[i] << "\n";
    }
}

int main() {
    int numVertices, numEdges, source;
    cout << "Enter the number of vertices: ";
    cin >> numVertices;

    cout << "Enter the number of edges: ";
    cin >> numEdges;

    vector<vector<Edge>> graph(numVertices);

    cout << "Enter the edges in the format 'source destination weight':\n";
    for (int i = 0; i < numEdges; ++i) {
        int u, v, weight;
        cin >> u >> v >> weight;
        graph[u].push_back({v, weight});
        // If the graph is undirected, uncomment the line below
        // graph[v].push_back({u, weight});
    }

    cout << "Enter the source vertex: ";
    cin >> source;

    dijkstra(graph, source);

    return 0;
}
```

Output

```
Enter the number of vertices: 5
Enter the number of edges: 7
Enter the edges in the format 'source destination weight':
0 1 4
0 2 2
1 2 1
1 3 5
2 3 8
2 4 10
3 4 2
Enter the source vertex: 0

Vertex  Distance from Source
0       0
1       3
2       2
3       6
4       8
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 09

DOP: **DOC:**

Title: Program For double linked list

```
#include <iostream>

using namespace std;


struct Node {
   int num;
   Node* next;
   Node* prev;
};


Node* frst, * cur, * after, * before;


void init()
{
   frst = NULL;
}


void add_node()
{
   Node* temp;
   int t, ch, pos, st, found = 0;
   while (true)
   {
      cout << "\n\tAdd Menu \n1. Add at start\n2. Add at End\n3. Add at intermediate position\n4. Exit submenu\nEnter choice: ";
```

```
cin >> ch;
switch (ch)
{
case 1:
   temp = new Node();
   cout << "\n Enter data: ";
   cin >> t;
   temp->num = t;
   temp->next = frst;
   temp->prev = NULL;
   frst = temp;
   break;
case 2:
   temp = new Node();
   cout << "\n Enter data: ";
   cin >> t;
   temp->num = t;
   cur = frst;
   if (cur == NULL)
   {
      frst = temp;
   }
   else
   {
      while (cur->next != NULL)
      {
         cur = cur->next;
      }
      cur->next = temp;
   }
```

```
    temp->next = NULL;
    temp->prev = cur;
    break;
case 3:
    cout << "\n Enter the position after which you want to enter data: ";
    cin >> pos;
    st = 1;
    cur = frst;
    while (cur->next != NULL)
    {
        if (st == pos)
        {
            found = 1;
            break;
        }
        else
        {
            cur = cur->next;
            st++;
        }
    }
    if (found == 1)
    {
        temp = new Node();
        cout << "\n Enter data: ";
        cin >> t;
        temp->num = t;
        after = cur->next;
        temp->next = cur->next;
        after->prev = temp;
```

```
            cur->next = temp;
            temp->prev = cur;
        }
        else
            cout << "\nPosition out of range";
        break;
    case 4:
        return;
    default:
        cout << "\n\n Wrong choice\n\n";
    }
  }
}
void del_node()
{
  Node* temp;
  int t, ch, pos, st, found = 0;
  while (true)
  {
    cout << "\n\tDelete Menu \n1. Delete starting node\n2. Delete End node\n3. Delete an intermediate node\n4. Exit submenu\nEnter choice: ";
    cin >> ch;

    switch (ch)
    {
    case 1:
      if (frst == NULL)
      {
        cout << "\n Empty list: nothing deleted";
        break;
      }
```

```
    temp = frst;
    frst = frst->next;
    frst->prev = NULL;
    cout << "\nNode deleted";
    delete temp;
    break;
case 2:
    if (frst == NULL)
    {
        cout << "\n Empty list: nothing deleted";
        break;
    }
    if (frst->next != NULL)
    {
        temp = frst;
        cur = before = frst;
        cur = cur->next;
        while (cur->next != NULL)
        {
            before = cur;
            cur = cur->next;
        }
        temp = cur;
        before->next = NULL;
        cout << "\nNode deleted";
        delete temp;
    }
    else
    {
        delete frst;
```

```
        frst = NULL;
        cout << "\nNode deleted";
    }
    break;
case 3:
    cout << "\n Enter the position after which you want to delete node: ";
    cin >> pos;
    st = 1;
    cur = frst;
    while (cur->next != NULL)
    {
        if (st == pos)
        {
            found = 1;
            break;
        }
        else
        {
            cur = cur->next;
            st++;
        }
    }
    if (found == 1)
    {
        temp = cur->next;
        after = temp->next;
        cur->next = after;
        after->prev = cur;
        cout << "\nNode deleted";
        delete temp;
```

```
            }
            else
                cout << "\nPosition out of range";
            break;
        case 4:
            return;
        default:
            cout << "\n\n Wrong choice\n\n";
        }
    }
}

void display()
{
    cur = frst;
    if (cur == NULL)
    {
        cout << "\nEmpty Link!";
        return;
    }
    cout << "\nDouble Link List contents are:\n";
    do
    {
        cout << cur->num << endl;
        cur = cur->next;
    } while (cur != NULL);
}

int main()
{
```

```
    int ch;
    init();
    while (true)
    {
        cout << "\n\tMain Menu \n1. Add element\n2. Delete element\n3. Display elements \n4.
Exit\nEnter choice: ";
        cin >> ch;
        switch (ch)
        {
        case 1:
            add_node();
            break;
        case 2:
            del_node();
            break;
        case 3:
            display();
            break;
        case 4:
            exit(0);
        default:
            cout << "\n\n Wrong choice\n\n";
        }
    }
    return 0;
}
```

Output

Main Menu

1. Add element

2. Delete element

3. Display elements

4. Exit

Enter choice : 1

Add Menu

1. Add at start

2. Add at End

3. Add at intermediate position

4. Exit submenu

Enter choice : 1

10

Enter choice : 3

10

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 10

DOP: **DOC:**

Title: Program For addition of two polynomials using link list.

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

struct term
{
   int coef;
   int order;
   struct term* next;
};
s
struct term* poly1 = nullptr;
struct term* poly2 = nullptr;
struct term* poly3 = nullptr;
struct term* cur = nullptr;
struct term* temp = nullptr;

void init()
{
   poly1 = nullptr;
   poly2 = nullptr;
   poly3 = nullptr;
}

void readpoly()
{
   int deg1, deg2, i, tcoef;
   cout << "\nEnter the degree of the first polynomial: ";
   cin >> deg1;

   for (i = deg1; i >= 0; i--)
   {
      cout << "\n\tEnter coefficient for x^" << i << " = ";
      cin >> tcoef;

      if (tcoef != 0)
      {
```

```
            temp = new term;
            temp->coef = tcoef;
            temp->order = i;
            temp->next = nullptr;

            cur = poly1;
            if (cur == nullptr)
               poly1 = temp;
            else
            {
               while (cur->next != nullptr)
                  cur = cur->next;
               cur->next = temp;
            }
         }
      }

      cout << "\nEnter the degree of the second polynomial: ";
      cin >> deg2;

      for (i = deg2; i >= 0; i--)
      {
         cout << "\n\tEnter value for x^" << i << " = ";
         cin >> tcoef;

         if (tcoef != 0)
         {
            temp = new term;
            temp->coef = tcoef;
            temp->order = i;
            temp->next = nullptr;

            cur = poly2;
            if (cur == nullptr)
               poly2 = temp;
            else
            {
               while (cur->next != nullptr)
                  cur = cur->next;
               cur->next = temp;
            }
         }
      }
   }

   void addpoly()
```

```
{
    struct term* curpoly1 = poly1;
    struct term* curpoly2 = poly2;

    while (curpoly1 != nullptr && curpoly2 != nullptr)
    {
        if (curpoly1->order == curpoly2->order)
        {
            temp = new term;
            temp->coef = curpoly1->coef + curpoly2->coef;
            temp->order = curpoly1->order;
            temp->next = nullptr;

            cur = poly3;
            if (cur == nullptr)
                poly3 = temp;
            else
            {
                while (cur->next != nullptr)
                    cur = cur->next;
                cur->next = temp;
            }

            curpoly1 = curpoly1->next;
            curpoly2 = curpoly2->next;
        }
        else if (curpoly1->order > curpoly2->order)
        {
            temp = new term;
            temp->coef = curpoly1->coef;
            temp->order = curpoly1->order;
            temp->next = nullptr;

            cur = poly3;
            if (cur == nullptr)
                poly3 = temp;
            else
            {
                while (cur->next != nullptr)
                    cur = cur->next;
                cur->next = temp;
            }

            curpoly1 = curpoly1->next;
        }
        else if (curpoly2->order > curpoly1->order)
```

```
        {
            temp = new term;
            temp->coef = curpoly2->coef;
            temp->order = curpoly2->order;
            temp->   temp->order = curpoly1->order;
        temp->next = nullptr;

        if (poly3->next == nullptr) {
            poly3->next = temp;
            cur = temp;
        } else {
            while (cur->next != nullptr) {
                cur = cur->next;
            }
            cur->next = temp;
            cur = temp;
        }

        curpoly1 = curpoly1->next;
    }

    while (curpoly2 != nullptr) {
        Node* temp = new Node();
        temp->coef = curpoly2->coef;
        temp->order = curpoly2->order;
        temp->next = nullptr;

        if (poly3->next == nullptr) {
            poly3->next = temp;
            cur = temp;
        } else {
            while (cur->next != nullptr) {
                cur = cur->next;
            }
            cur->next = temp;
            cur = temp;
        }

        curpoly2 = curpoly2->next;
    }
}

void displayResult() {
    std::cout << "The first polynomial is:" << std::endl;
    cur = poly1->next;
    while (cur != nullptr) {
```

```
        std::cout << "(" << cur->coef << ")x^" << cur->order;
        if (cur->next != nullptr) {
            std::cout << " + ";
        }
        cur = cur->next;
    }
    std::cout <<  = " 0\n";
}

int main() {
    system("clear"); // Clear the screen

    init();
    std::cout << "\nEnter the two polynomials : \n";
    readpoly();
    system("clear"); // Clear the screen
    addpoly();
    std::cout << "\nResult : \n";
    displayResult();

    getchar(); // Wait for user input

    return 0;
}
```

Output

```
/*
Enter the degree of the frst polynomial :03
Enter coefcient for x^3 = 3
Enter coefcient for x^2 = 3
Enter coefcient for x^1 = 10
Enter coefcient for x^0 = 10
Enter the degree of the second polynomial :03
Enter coefcient for x^3 = 3
Enter coefcient for x^2 = 3
Enter coefcient for x^1 = 10
Enter coefcient for x^0 = 10
The First polynomial is
3 x^3 +3 x^2 + 10 x^1 +10 x^0
The second polynomial is
3 x^3 +3 x^2 + 10 x^1 +10 x^0
Addition of polynomial is
6 x^3 +6 x^2 + 20 x^1 +20 x^0
*/
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 11

DOP: **DOC:**

Title: Program for binary tree traversal using non-recursive methods

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

struct node
{
   int data;
   node* left;
   node* right;
};

void inorderTraversal(node* root)
{
   if (root == nullptr)
      return;

   stack<node*> stack;
   node* curr = root;

   while (curr != nullptr || !stack.empty())
   {
      while (curr != nullptr)
      {
         stack.push(curr);
         curr = curr->left;
      }

      curr = stack.top();
      stack.pop();

      cout << curr->data << " ";

      curr = curr->right;
   }
```

```
}

void preorderTraversal(node* root)
{
    if (root == nullptr)
        return;

    stack<node*> stack;
    stack.push(root);

    while (!stack.empty())
    {
        node* curr = stack.top();
        stack.pop();

        cout << curr->data << " ";

        if (curr->right != nullptr)
            stack.push(curr->right);

        if (curr->left != nullptr)
            stack.push(curr->left);
    }
}

void postorderTraversal(node* root)
{
    if (root == nullptr)
        return;

    stack<node*> stack;
    stack.push(root);
    stack<node*> result;

    while (!stack.empty())
    {
        node* curr = stack.top();
        stack.pop();

        result.push(curr);

        if (curr->left != nullptr)
            stack.push(curr->left);
```

```
        if (curr->right != nullptr)
            stack.push(curr->right);
    }

    while (!result.empty())
    {
        node* curr = result.top();
        result.pop();

        cout << curr->data << " ";
    }
}

int main()
{
    // Create a binary tree
    node* root = new node();
    root->data = 1;
    root->left = new node();
    root->left->data = 2;
    root->right = new node();
    root->right->data = 3;
    root->left->left = new node();
    root->left->left->data = 4;
    root->left->right = new node();
    root->left->right->data = 5;

    cout << "Inorder Traversal: ";
    inorderTraversal(root);

    cout << "\nPreorder Traversal: ";
    preorderTraversal(root);

    cout << "\nPostorder Traversal: ";
    postorderTraversal(root);

    return 0;
}
```

Output

```
Inorder Traversal: 4 2 5 1 3
Preorder Traversal: 1 2 4 5 3
Postorder Traversal: 4 5 2 3 1
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 12

DOP: **DOC:**

Title: Program for max heap

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

void maxHeapify(int heap[], int n, int i)
{
   int largest = i;
   int left = 2 * i + 1;
   int right = 2 * i + 2;

   if (left < n && heap[left] > heap[largest])
      largest = left;

   if (right < n && heap[right] > heap[largest])
      largest = right;

   if (largest != i)
   {
      swap(heap[i], heap[largest]);
      maxHeapify(heap, n, largest);
   }
}

void buildMaxHeap(int heap[], int n)
{
   for (int i = n / 2 - 1; i >= 0; i--)
      maxHeapify(heap, n, i);
}

void heapSort(int heap[], int n)
{
   buildMaxHeap(heap, n);

   for (int i = n - 1; i > 0; i--)
   {
      swap(heap[0], heap[i]);
      maxHeapify(heap, i, 0);
```

```
    }
}

int main()
{
    int heap[10], no;

    cout << "Enter the number of elements: ";
    cin >> no;

    cout << "Enter the numbers: ";
    for (int i = 0; i < no; i++)
        cin >> heap[i];

    heapSort(heap, no);

    cout << "Sorted array: ";
    for (int i = 0; i < no; i++)
        cout << heap[i] << " ";

    cout << endl;

    return 0;
}
```

Output

```
/*
Enter no of elements :6
Enter the nos :
12
34
56
1
78
90
Heap array : 90 56 78 1
 12 34
The sorted array is : 1 12 34 56 78 90
*/
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 13

DOP: **DOC:**

Title: Program for min heap

```
#include <iostream>
#include <limits>

/* Declaring heap globally so that we do not need to pass it as an argument every time */
/* Heap implemented here is Min Heap */
int heap[1000000], heapSize;

/* Initialize Heap */
void Init()
{
   heapSize = 0;
   heap[0] = -std::numeric_limits<int>::max();
}

/* Insert an element into the heap */
void Insert(int element)
{
   heapSize++;
   heap[heapSize] = element; /* Insert in the last place */
   /* Adjust its position */
   int now = heapSize;
   while (heap[now / 2] > element)
   {
      heap[now] = heap[now / 2];
      now /= 2;
   }
   heap[now] = element;
}

int DeleteMin()
{
   /* heap[1] is the minimum element. So we remove heap[1]. Size of the heap is decreased.
   Now heap[1] has to be filled. We put the last element in its place and see if it fits.
   If it does not fit, take the minimum element among both its children and replace the parent
with it.
```

```
    Again see if the last element fits in that place. */
    int minElement, lastElement, child, now;
    minElement = heap[1];
    lastElement = heap[heapSize--];
    /* now refers to the index at which we are now */

    for (now = 1; now * 2 <= heapSize; now = child)
    {
        /* child is the index of the element which is minimum among both the children */
        /* Indexes of children are i*2 and i*2 + 1 */
        child = now * 2;
        /* child != heapSize because heap[heapSize + 1] does not exist, which means it has only
one child */
        if (child != heapSize && heap[child + 1] < heap[child])
        {
            child++;
        }
        /* To check if the last element fits or not, it suffices to check if the last element is less
than the minimum element among both the children */
        if (lastElement > heap[child])
        {
            heap[now] = heap[child];
        }
        else /* It fits there */
        {
            break;
        }
    }
    heap[now] = lastElement;
    return minElement;
}

int main()
{
    int number_of_elements;
    std::cout << "Program to demonstrate Heap:\nEnter the number of elements: ";
    std::cin >> number_of_elements;
    int iter, element;
    Init();
    std::cout << "Enter the elements: ";
    for (iter = 0; iter < number_of_elements; iter++)
    {
        std::cin >> element;
        Insert(element);
    }
    for (iter = 0; iter < number_of_elements; iter++)
```

```
    {
        std::cout << DeleteMin() << " ";
    }
    std::cout << "\n";
    return 0;
}
```

Output

```
/*
Program to demonstrate Heap:
Enter the number of elements: 6
Enter the elements: 12
2
34
67
89
88
2 12 34 67 88 89
*/
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 14

DOP: **DOC:**

Title: Program for Dijkstra Shortest path algo

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

int a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;

void bfs(int v)
{
   for (i = 1; i <= n; i++)
   {
      if (a[v][i] && !visited[i])
         q[++r] = i;
   }
   if (f <= r)
   {
      visited[q[f]] = 1;
      bfs(q[f++]);
   }
}

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

   for (i = 1; i <= n; i++)
   {
      q[i] = 0;
      visited[i] = 0;
   }

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

```
    }
  }

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

  bfs(v);

  cout << "\n The nodes that are reachable are:\n";
  for (i = 1; i <= n; i++)
  {
    if (visited[i])
      cout << i << "\t";
    else
    {
      cout << "\n Bfs is not possible";
      break;
    }
  }

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

Output

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

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 15

DOP: **DOC:**

Title: Program for kruskal algorithm

```
#include <iostream>
#include <limits>
#define V 9

int minDistance(int dist[], bool sptSet[])
{
   int min = INT_MAX, min_index;
   for (int v = 0; v < V; v++)
      if (sptSet[v] == false && dist[v] <= min)
         min = dist[v], min_index = v;
   return min_index;
}

void printSolution(int dist[], int n)
{
   std::cout << "Vertex Distance from Source\n";
   for (int i = 0; i < V; i++)
      std::cout << i << "\t" << dist[i] << "\n";
}

void dijkstra(int graph[V][V], int src)
{
   int dist[V];
   bool sptSet[V];
   for (int i = 0; i < V; i++)
      dist[i] = INT_MAX, sptSet[i] = false;
   dist[src] = 0;
   for (int count = 0; count < V - 1; count++)
   {
      int u = minDistance(dist, sptSet);
      sptSet[u] = true;
      for (int v = 0; v < V; v++)
         if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] <
dist[v])
```

```
            dist[v] = dist[u] + graph[u][v];
    }
    printSolution(dist, V);
}

int main()
{
    int graph[V][V] = {{0, 6, 0, 0, 0, 0, 0, 8, 0},
                       {6, 0, 8, 0, 0, 0, 0, 13, 0},
                       {0, 8, 0, 7, 0, 6, 0, 0, 2},
                       {0, 0, 7, 0, 9, 14, 0, 0, 0},
                       {0, 0, 0, 9, 0, 10, 0, 0, 0},
                       {0, 0, 6, 14, 10, 0, 2, 0, 0},
                       {0, 0, 0, 0, 0, 2, 0, 1, 6},
                       {8, 13, 0, 0, 0, 0, 1, 0, 7},
                       {0, 0, 2, 0, 0, 0, 6, 7, 0}};
    dijkstra(graph, 0);
    return 0;
}
```

Output

```
Vertex Distance from Source
0       0
1       6
2       14
3       15
4       22
5       12
6       18
7       8
8       15
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 16

DOP: **DOC:**

Title: Program for Prims Algorithm

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

int main()
{
int a, b, n, ne = 1, i, j, min, mincost = 0;
cout << "\n Enter The number of Vertices: ";
cin >> n;
int cost[10][10];
cout << "\n Enter The adj Matrix\n";
for (i = 1; i <= n; i++)
{
for (j = 1; j <= n; j++)
{
cin >> cost[i][j];
if (cost[i][j] == 0)
{
cost[i][j] = 999;
}
}
}
while (ne < n)
{
min = 999;
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;
}
}
}
cout << "edge(" << a << "," << b << ")=" << min << endl;
mincost = mincost + min;
```

```
c
Copy code
   cost[a][b] = cost[b][a] = 999;
   ne++;
}
cout << "\nMinimum spanning Tree of weight=" << mincost << endl;
return 0;
}
```

Output

```
Enter the no of Vertices=4
Enter the adjecent Matrix=
0 7 1 6
7 0 5 2
1 5 0 3
6 2 3 0
Edge(1,3)=1
Edge(2,4)=2
Edge(3,4)=3
Minimum spanning Tree of wt=6
*/
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Practical: 17

DOP: **DOC:**

Title: Program for Hash Table using linear probing without replacement

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

int main()
{
   int a, b, n, ne = 1, i, j, min, mincost = 0, visited[10] = {0};
   cout << "\n Enter The number of Vertices: ";
   cin >> n;
   int cost[10][10];
   cout << "\n Enter The adj Matrix\n";
   for (i = 1; i <= n; i++)
   {
      for (j = 1; j <= n; j++)
      {
         cin >> cost[i][j];
         if (cost[i][j] == 0)
         {
            cost[i][j] = 999;
         }
      }
   }
   visited[1] = 1;
   while (ne < n)
   {
```

```
    min = 999;
    for (i = 1; i <= n; i++)
    {
      for (j = 1; j <= n; j++)
      {
        if (visited[i] == 1)
        {
          if (cost[i][j] < min)
          {
            min = cost[i][j];
            a = i;
            b = j;
          }
        }
      }
    }
    if (visited[a] == 0 || visited[b] == 0)
    {
      cout << "edge(" << a << "," << b << ")=" << min << endl;
      mincost = mincost + min;
      cost[a][b] = cost[b][a] = 999;
      ne++;
      visited[b] = 1;
    }
  }
  cout << "\nMinimum spanning Tree of weight=" << mincost << endl;
  return 0;
}
```

Output

Enter The no of Vertices=4

Enter The adj Matrix

0 7 1 6

7 0 5 2

1 5 0 3

6 2 3 0

edge(1,3)=1

edge(3,4)=3

edge(4,2)=2

Minimum spanning Tree of wt=6

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

=

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

Practical: 18

DOP: **DOC:**

Title: Program for binary search

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

int count = 0;

struct HashTable
{
   int data;
   int delStatus; /* 0 means valid data present, 1 means data is not present in the bucket */
};

HashTable ht[size];

void initHT()
{
   for (int i = 0; i < size; i++)
      ht[i].delStatus = 1;
}

void addData()
{
   int data, key, i;
   if (count == size)
   {
      std::cout << "\nHash Table FULL\n";
   }
   else
   {
      std::cout << "\nEnter data: ";
      std::cin >> data;
      key = data % size; /Applying hash function/
      if (ht[key].delStatus == 1)
      {
         ht[key].data = data;
         ht[key].delStatus = 0;
         std::cout << "\n" << data << " is added to the hash table\n";
         count++;
      }
      else
```

```
        {
            for (i = 0; i < size; i++)
            {
                key = (key + 1) % size;
                if (ht[key].delStatus == 1)
                {
                    ht[key].data = data;
                    ht[key].delStatus = 0;
                    std::cout << "\nData Added to table\n";
                    count++;
                    break;
                }
            }
        }
    }
}

void delData()
{
    int data, key, i, flag = 0;
    if (count == 0)
    {
        std::cout << "\nHash Table EMPTY\n";
    }
    else
    {
        std::cout << "\nEnter data to delete: ";
        std::cin >> data;
        key = data % size; /Applying hash function/
        if (ht[key].delStatus == 0 && ht[key].data == data)
        {
            ht[key].delStatus = 1;
            std::cout << "\n" << data << " is deleted from the table\n";
            count--;
        }
        else
        {
            for (i = 0; i < size; i++)
            {
                key = (key + 1) % size;
                if (ht[key].data == data)
                {
                    ht[key].delStatus = 1;
                    std::cout << "\nData deleted from the table\n";
                    count--;
                    flag = 1;
                    break;
                }
            }
            if (flag == 0)
```

```
            std::cout << "\nData to be deleted not found in the table\n";
        }
    }
}

void display()
{
    for (int i = 0; i < size; i++)
    {
        if (ht[i].delStatus == 0)
        {
            std::cout << "| " << ht[i].data << " |";
        }
        else
            std::cout << "| |";
    }
    std::cout << std::endl;
}

int main()
{
    int ch;
    initHT();
    do
    {
        std::cout << "\nMain menu:\n";
        std::cout << "\n1) Add data \n2) Delete data \n3) Display table \n4) Exit\n\nEnter
choice: ";
        std::cin >> ch;
        switch (ch)
        {
        case 1:
            addData();
            break;
        case 2:
            delData();
            break;

        case 3:
            display();
            break;

        case 4:
            exit(0);
            break;

        default:
            std::cout("\nWrong choice!!");
 }
 }while(ch != 4);
```

}

Output

```
/*
Main menu :
1) Add data
2) Delete data
3) Display table
4) Exit
Enter choice :1
Enter data : 10
10 is added to hash table
Main menu :
1) Add data
2) Delete data
3) Display table
4) Exit
Enter choice :1
Enter data : 20
Data Added to table
Main menu :
1) Add data
2) Delete data
3) Display table
4) Exit
Enter choice :1
Enter data : 40
Data Added to table
Main menu :
1) Add data
2) Delete data
3) Display table
4) Exit
Enter choice :3
| 10 || 20 || 40 || || || || || || || |
Main menu :
1) Add data
2) Delete data
3) Display table
4) Exit
Enter choice :2
Enter data to delete : 20
Data deleted from table
Main menu :
1) Add data
2) Delete data
3) Display table
4) Exit
Enter choice :3
```

```
| 10 || || 40 || || || || || || || |
Main menu :
1) Add data
2) Delete data
3) Display table
4) Exit
Enter choice :1
Enter data : 04
4 is added to hash table
Main menu :
1) Add data
2) Delete data
3) Display table
4) Exit
Enter choice :3
| 10 || || 40 || || 4 || || || || || |
Main menu :
1) Add data
2) Delete data
3) Display table
4) Exit
Enter choice :
*/
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 19

DOP: **DOC:**

Title: Program for linear search

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

int main()
{
   int i, low, high, mid, n, key, array[100];
   cout << "Enter number of elements: ";
   cin >> n;
   cout << "Enter " << n << " integers: ";
   for (i = 0; i < n; i++)
      cin >> array[i];
   cout << "Enter value to find: ";
   cin >> key;
   low = 0;
   high = n - 1;
   mid = (low + high) / 2;
   while (low <= high)
   {
      if (array[mid] < key)
         low = mid + 1;
      else if (array[mid] == key)
      {
         cout << key << " found at location " << mid + 1 << endl;
         break;
      }
      else
         high = mid - 1;
      mid = (low + high) / 2;
   }
   if (low > high)
      cout << "Not found! " << key << " isn't present in the list." << endl;
   return 0;
}
```

Output

/*
Enter number of elementsn7
Enter 7 integersn
10
20
30
40
50
60
70
Enter value to fndn40
40 found at location 4

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 20

DOP: **DOC:**

Title:

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

int main()
{
   int array[50], i, target, num;
   cout << "How many elements do you want in the array: ";
   cin >> num;
   cout << "Enter array elements: ";
   for (i = 0; i < num; ++i)
      cin >> array[i];
   cout << "Enter element to search: ";
   cin >> target;
   for (i = 0; i < num; ++i)
   {
      if (array[i] == target)
         break;
   }
   if (i < num)
      cout << "Target element found at location " << i << endl;
   else
      cout << "Target element not found in the array" << endl;
   return 0;
}
```

Output

```
/*
How many elements do you want in the array5
Enter array elements:
10
20
30
```

```
40
50
Enter element to search:30
Target element found at location 2
*/
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 21

DOP: **DOC:**

Title : Program for bubble sort

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

void bubble_sort(int a[], int n)
{
   int i = 0, j = 0, tmp;
   for (i = 0; i < n; i++)
   {
      // loop n times - 1 per element
      for (j = 0; j < n - i - 1; j++)
      {
         // last i elements are sorted already
         if (a[j] > a[j + 1])
         {
            // swap if order is broken
            tmp = a[j];
            a[j] = a[j + 1];
            a[j + 1] = tmp;
         }
      }
   }
}

int main()
{
   int a[100], n, i, d, swap;
   cout << "Enter number of elements in the array: ";
   cin >> n;
   cout << "Enter Array elements: " << endl;
   for (i = 0; i < n; i++)
      cin >> a[i];
   bubble_sort(a, n);
   cout << "Printing the sorted array: " << endl;
```

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

Output

```
Enter number of elements in the array: 5
Enter Array elements:
9
3
7

1
5
Printing the sorted array:
1
3
5
7
9
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 22

DOP: **DOC:**

Title : Program for insertion sort

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

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

int main()
{
   int a[100], n, k, i, j, temp;
   cout << "Enter the number of elements: ";
   cin >> n;
   cout << "Enter the elements of the array: " << endl;
   for (i = 0; i < n; i++)
   {
      cin >> a[i];
   }
   insertion_sort(a, n);
   cout << "Elements of the array after sorting are: " << endl;
   for (i = 0; i < n; i++)
   {
      cout << a[i] << endl;
```

```
    }
    return 0;
}
```

Output

```
how many elements7
enter the elements of aaray12 44 54 34 66 77 1
elements of array after sorting are :
1
12
34
44
54
66
77
*/
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 23

DOP: **DOC:**

Title : Program for radix sort

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

int getMax(int arr[], int n)
{
   int mx = arr[0];
   for (int i = 1; i < n; i++)
   {
      if (arr[i] > mx)
         mx = arr[i];
   }
   return mx;
}

void countSort(int arr[], int n, int exp)
{
   int output[n];
   int count[10] = {0};

   for (int i = 0; i < n; i++)
      count[(arr[i] / exp) % 10]++;

   for (int i = 1; i < 10; i++)
      count[i] += count[i - 1];

   for (int i = n - 1; i >= 0; i--)
   {
      output[count[(arr[i] / exp) % 10] - 1] = arr[i];
      count[(arr[i] / exp) % 10]--;
   }

   for (int i = 0; i < n; i++)
      arr[i] = output[i];
}
```

```
void radixsort(int arr[], int n)
{
   int m = getMax(arr, n);

   for (int exp = 1; m / exp > 0; exp *= 10)
      countSort(arr, n, exp);
}

void print(int arr[], int n)
{
   cout << "\nSorted array is: ";
   for (int i = 0; i < n; i++)
      cout << arr[i] << " ";
   cout << endl;
}

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

   int arr[100];
   cout << "Enter the array elements: " << endl;
   for (int i = 0; i < n; i++)
   {
      cout << "Enter element " << i + 1 << ": ";
      cin >> arr[i];
   }

   radixsort(arr, n);

   print(arr, n);

   return 0;
}
```

Output

```
/*
enter size of array 7
enter array element
enter element 2
```

```
enter element 3
enter element 1
enter element 7
enter element 5
enter element 9
enter element 11
sorted array is 1 2 3 5 7 9 11
*/
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 24

DOP: **DOC:**

Title : Program for Quick sort

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

void qsort(int arr[], int low, int high);
int partition(int arr[], int low, int high);

int main()
{
   int arr[50], n;
   cout << "Enter the size of the array: ";
   cin >> n;

   cout << "Enter array elements:" << endl;
   for (int i = 0; i < n; i++)
   {
      cout << "Enter element " << i + 1 << ": ";
      cin >> arr[i];
   }

   qsort(arr, 0, n - 1);

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

   return 0;
}

void qsort(int arr[], int low, int high)
{
   int j;
   if (low < high)
```

```
    {
        j = partition(arr, low, high);
        qsort(arr, low, j - 1);
        qsort(arr, j + 1, high);
    }
}

int partition(int arr[], int low, int high)
{
    int pivot = arr[low];
    int i = low + 1;
    int j = high;

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

        if (i > j)
            break;

        // Swap arr[i] and arr[j]
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

    // Swap pivot (arr[low]) and arr[j]
    int temp = arr[low];
    arr[low] = arr[j];
    arr[j] = temp;

    return j;
}
```

Output

```
Enter size of array:7
Enter array elements:
enter 1 element:12
enter 2 element:1
enter 3 element:56
```

```
enter 4 element:78
enter 5 element:22
enter 6 element:68
enter 7 element:89
Sorted array is:
1

12

22

56

68

78

89

*/
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 25

DOP: **DOC:**

Title : Program for merge sort

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

void merge(int arr[], int min, int mid, int max);
void part(int arr[], int min, int max);

int main()
{
   int arr[30];
   int size;
   cout << "------- Merge sorting method -------" << endl << endl;
   cout << "Enter total number of elements: ";
   cin >> size;

   for (int i = 0; i < size; i++)
   {
      cout << "Enter element " << i + 1 << ": ";
      cin >> arr[i];
   }

   part(arr, 0, size - 1);

   cout << endl << "------- Merge sorted elements -------" << endl << endl;
   for (int i = 0; i < size; i++)
      cout << arr[i] << " ";
   cout << endl;

   return 0;
}

void part(int arr[], int min, int max)
{
   int mid;
   if (min < max)
```

```
    {
        mid = (min + max) / 2;
        part(arr, min, mid);
        part(arr, mid + 1, max);
        merge(arr, min, mid, max);
    }
}

void merge(int arr[], int min, int mid, int max)
{
    int tmp[30];
    int i, j, k;
    j = min;
    k = min;
    int m = mid + 1;

    while (j <= mid && m <= max)
    {
        if (arr[j] <= arr[m])
        {
            tmp[k] = arr[j];
            j++;
        }
        else
        {
            tmp[k] = arr[m];
            m++;
        }
        k++;
    }

    while (j <= mid)
    {
        tmp[k] = arr[j];
        j++;
        k++;
    }

    while (m <= max)
    {
        tmp[k] = arr[m];
        m++;
        k++;
    }
```

```
    for (i = min; i <= max; i++)
        arr[i] = tmp[i];
}

Output

Enter total no. of elements : 05
Enter 1 element : 23
Enter 2 element : 12
Enter 3 element : 34
Enter 4 element : 1
Enter 5 element : 33
------- Merge sorted elements -------
1 12 23 33 34
*/
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.

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

Department of Computer Applications

Practical: 26

DOP: **DOC:**

Title : Program for heap sort

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

void create(int[]);
void down_adjust(int[], int);

int main()
{
   int heap[30], n, i, last, temp;
   cout << "Enter the number of elements: ";
   cin >> n;
   cout << "Enter the elements: ";
   for (i = 1; i <= n; i++)
      cin >> heap[i];

   // Create a heap
   heap[0] = n;
   create(heap);

   // Sorting
   while (heap[0] > 1)
   {
      // Swap heap[1] and heap[last]
      last = heap[0];
      temp = heap[1];
      heap[1] = heap[last];
      heap[last] = temp;
      heap[0]--;
      down_adjust(heap, 1);
   }

   // Print sorted data
   cout << "\nArray after sorting:" << endl;
   for (i = 1; i <= n; i++)
```

```
        cout << heap[i] << " ";
    cout << endl;

    return 0;
}

void create(int heap[])
{
    int i, n;
    n = heap[0]; // Number of elements
    for (i = n / 2; i >= 1; i--)
        down_adjust(heap, i);
}

void down_adjust(int heap[], int i)
{
    int j, temp, n, flag = 1;
    n = heap[0];

    while (2 * i <= n && flag == 1)
    {
        j = 2 * i; // j points to left child
        if (j + 1 <= n && heap[j + 1] > heap[j])
            j = j + 1;
        if (heap[i] > heap[j])
            flag = 0;
        else
        {
            temp = heap[i];
            heap[i] = heap[j];
            heap[j] = temp;
            i = j;
        }
    }
}
```

Output

```
/*
Enter no. of elements:6
Enter elements:67
12
1
45
```

```
78
23
Array after sorting:
1 12 23 45 67 78
*/
```

Submitted By **Checked By :**

Sign :

Name :

<Name of Faculty>MCA-1

Roll No.