# DEPARTMENT OF COMPUTER SCIENCE

**<u>RECORD NOTE</u>**

Record work submitted to the Bharathiar University in partial fulfillment of the requirement for the Degree of

# Master of Science in Computer Science

**PROGRAMMING LAB – Algorithm and OOPS Lab**

**SHRI NEHRU MAHA VIDYALAYA COLLEGE OF ARTS & SCIENCE**
(Affiliated to Bharathiar University)

Shri Gambhirmal Bafna Nagar

Malumachampatti, Coimbatore-641050

**OCTOBER – 2024**

SHRI NEHRU MAHA VIDYALAYA
COLLEGE OF ARTS AND SCIENCE
SNMV
(Affiliated to Bharathiar University, Coimbatore, Re-accredited with "A" Grade by NAAC)
Shri Gambhirmal Bafna Nagar, Malumachampatti, Coimbatore - 641 050, Tamil Nadu, India.
Estd.: 1989
Estd.: 1964

# MASTER OF SCIENCE IN COMPUTER SCIENCE

This is to certify that it is a bonafide record work done by _________________________ Studying I year M.Sc . Computer Science

Reg .No. ________________________

__________________ ____________________

**Staff In – Charge** **Head of the Department**

**Submitted for**: **Practical I : Algorithm and OOPS Lab**

Bharathiar University Practical Examination held on __________________

Odd semester (2024-2025)

**Staff In-Charge**

| | |
|---|---|
| EX.NO: 1 | **TOWER OF HANOAI USING RECURSION** |
| DATE: | |

**AIM:**

**ALGORITHM:**

**PROGRAM:**

```
#include <iostream>

using namespace std;

void towers(int num, char frompeg, char topeg, char auxpeg) {

 if (num == 1) {

cout << "\nMove disk 1 from peg " << frompeg << " to peg " << topeg;

 return;

} towers(num - 1, frompeg, auxpeg, topeg);

 cout << "\nMove disk " << num << " from peg " << frompeg << " to peg " << topeg;

towers(num - 1, auxpeg, topeg, frompeg);

 }

int main() {

int num;

 cout << "Enter the number of disks: ";

cin >> num;

cout << "\nThe sequence of moves involved in Tower of Hanoi is:";

 towers(num, 'A', 'C', 'B');

cout << endl;

return 0;
}
```

**OUTPUT:**

```
Enter the number of disks: 3

The sequence of moves involved in Tower of Hanoi is:

 Move disk 1 from peg A to peg C

Move disk 2 from peg A to peg B

 Move disk 1 from peg C to peg B

 Move disk 3 from peg A to peg C

 Move disk 1 from peg B to peg A

 Move disk 2 from peg B to peg C

 Move disk 1 from peg A to peg C
```

**RESULT:**

|  |  |
|---|---|
| EX.NO: 2 | **BINARY SEARCH TREE USING TRAVERSE** |
| DATE: |  |

**AIM:**

.

**ALGORITHM:**

**PROGRAM:**

```
#include<iostream>
 #include<conio.h>
struct tree {
tree *left, *right;
int data;
 };
 tree *root = NULL;
 void create(tree *&root) {
int value;
char ch;
 if (root == NULL) {
root = new tree;
 std::cout << "\n Enter the value of the root node: ";
std::cin >> root->data;
 root->left = root->right = NULL;
}
 do {
 tree *p = root;
std::cout << "\n Enter the value of the node: ";
 std::cin >> value;
while (p) {
 if (value < p->data)
 { if (p->left == NULL) {
 p->left = new tree;
p = p->left;
 p->data = value;
 p->left = p->right = NULL;
std::cout << "\n Value entered in left";
break;
} else {
p = p->left;
 }
 } else if (value > p->data) {
```

```
if (p->right == NULL) {
 p->right = new tree;
 p = p->right;
 p->data = value;
p->left = p->right = NULL;
std::cout << "\n Value entered in right";
break;
 } else {

p = p->right;
 }
 } else {
 std::cout << "\n Duplicate value encountered. Ignoring.\n";
 break;
}
}
 std::cout << "\n Do you want to continue (y/n)? ";
 std::cin >> ch;
 } while (ch == 'y' || ch == 'Y');
}
void inorder(tree *p)
 { if (p != NULL) {
 inorder(p->left);
std::cout << p->data << " ";
inorder(p->right);
}
 } void preorder(tree *p) {
if (p != NULL) {
std::cout << p->data << " ";
 preorder(p->left);
preorder(p->right);
}
 }
void postorder(tree *p)
{ if (p != NULL) {
```

```
postorder(p->left);
postorder(p->right);
std::cout << p->data << " ";
 }
 }
 int main() {
create(root);
std::cout << "\n Printing traversal in inorder: ";
inorder(root);
 std::cout << "\n Printing traversal in preorder: ";
 preorder(root);
 std::cout << "\n Printing traversal in postorder: ";
postorder(root);
getch();
 return 0;
```

**OUTPUT:**

```
Enter the value of the root node: 25
 Enter the value of the node: 23
Value entered in left
Do you want to continue (y/n)? Y
 Enter the value of the node: 20
Value entered in left
Do you want to continue (y/n)? N
Printing traversal in inorder: 20 23 25
Printing traversal in preorder: 25 23 20
Printing traversal in postorder: 20 23 25
```

**RESULT :**

| | |
|---|---|
| EX.NO: 3 | **STACK USING LINKED LIST** |
| DATE: | |

**AIM:**

**ALGORITHM:**

.

**PROGRAM :**

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

 class Stack {
 private:
struct Node {
 int info;
 Node* next;
 };
Node* top;

 public:
 Stack() {
top = NULL;
 }

void push(int n);
void pop();
void peek();
 };
 void Stack::push(int n) {
Node* newNode = new Node;
 newNode->info = n;
 newNode->next = top;
top = newNode;
}

void Stack::pop() {
 if (top == NULL) {
cout << "\nStack empty";
 } else {
Node* temp = top;
top = top->next;
```

```
cout << "\nDeleted element is " << temp->info;
delete temp;
}
}
void Stack::peek()
{
Node* current = top;
if (current == NULL) {
cout << "\nStack is empty";
 } else {
cout << "\nStack elements:\n";
            while (current != NULL) {
           cout << current->info << "\n";
           current = current->next;
           }
           }
            }

            int main() {
            Stack obj;
            char choice;
            int ch, data;

            do {
           cout << "\nStack Operations";
           cout << "\n***************";
            cout << "\n1. Push";
            cout << "\n2. Pop";
           cout << "\n3. Peek";
           cout << "\n4. Exit";
           cout << "\nWhich operation do you want to perform? ";
            cin >> ch;

            switch (ch) {
           case 1:
```

```
cout << "\nEnter the data you want to push: ";
cin >> data;
obj.push(data);
break;
 case 2: obj.pop();
break;
 case 3:
obj.peek();
break;
case 4:
 exit(0);
break;
default:
cout << "\nPlease enter a valid choice";
}
cout << "\nDo you want to continue? (y/n): ";
cin >> choice;
} while (choice == 'y' || choice == 'Y');

return 0; // Standard return for main function
}
```

**OUTPUT:**

```
Stack Operations
 *************
1. Push
 2. Pop
 3. Peep
 4. Exit
Which operation do you want to perform? 1
Enter the data you want to push 3
Do you want to continue? y
Stack Operations
*************
1. Push
 2. Pop
3. Peep
4. Exit
 Which operation do you want to perform? 1
Enter the data you want to push 6
 Do you want to continue? y
Stack Operations
*************
1. Push
2. Pop
3. Peep
4. Exit
 Which operation do you want to perform? 1
Enter the data you want to push 9
 Do you want to continue? Y
 Stack Operations
 *************
 1. Push
2. Pop
```

```
 3. Peep
 4. Exit
Which operation do you want to perform? 3
 9
 6
 3
 Do you want to continue? Y
 Stack Operations
*************
1. Push
 2. Pop
3. Peep
4. Exit
 Which operation do you want to perform? 2
 Deleted element is 9
Do you want to continue? N
```

**RESULT:**

|  |  |
|---|---|
| EX.NO: 4 | **CIRCULAR QUEUE** |
| DATE: |  |

**AIM:**

**ALGORITHM:**

**PROGRAM :**

```
#include<iostream>
 #include<cstdlib>

 #define MAX 5
 using namespace std;
 class circular_queue {
 private:
 int *cqueue_arr;
 int front, rear;

 public: circular_queue() {
 cqueue_arr = new int[MAX];
rear = front = -1;
 }

 void insert(int item) {
 if ((front == 0 && rear == MAX - 1) || (front == rear + 1)) {
 cout << "Queue overflow \n";
 return;
 }

 if (front == -1) {
 front = 0;
 rear = 0;
} else {
 if (rear == MAX - 1)
rear = 0;
 else
 rear = rear + 1;
 }
cqueue_arr[rear] = item;
 }
 void del() {
```

```
 if (front == -1) {
 cout << "Queue underflow \n";
 return;
 }
 cout << "Element deleted from queue is: " << cqueue_arr[front] << "\n";
 if (front == rear) {
front = -1;
 rear = -1;
 } else {
 if (front == MAX - 1)
 front = 0;
 else
 front = front + 1;
 }
 }
 void display() {
 if (front == -1) {
 cout << "Queue is empty \n";
return;
 }
 cout << "Queue elements: ";
 int front_pos = front, rear_pos = rear;
 if (front_pos <= rear_pos) {
 while (front_pos <= rear_pos) {
 cout << cqueue_arr[front_pos] << " "; front_pos++; }
 } else {
 while (front_pos <= MAX - 1) {
 cout << cqueue_arr[front_pos] << " ";
front_pos++;
 }
front_pos = 0;
 while (front_pos <= rear_pos) {
cout << cqueue_arr[front_pos] << " ";
 front_pos++;
}
```

```
 }
 cout << endl;
 }
 };
 int main() {
 int choice, item;
 circular_queue cq;
 do {
cout << "\n1. Insert";
cout << "\n2. Delete";
 cout << "\n3. Display";
 cout << "\n4. Exit";
 cout << "\nEnter your choice: ";
 cin >> choice;

 switch (choice) {
case 1:
cout << "\nInput the element for insertion in queue: ";
cin >> item;
cq.insert(item);
 break;
 case 2:
 cq.del();
 break;
 case 3:
 cq.display();
 break;
 case 4:
 exit(0);
 break;
 default:
 cout << "\nWrong choice";
 }
 } while (choice != 4);
 return 0;
}
```

**OUTPUT:**

```
1. Insert
 2. Delete
 3. Display
 4. Exit
 Enter your choice: 1
 Input the elements for insertion in queue: 2
1. Insert
 2. Delete
3. Display
4. Exit
 Enter your choice: 1
 Input the elements for insertion in queue: 4
1. Insert
 2. Delete
 3. Display
 4. Exit
 Enter your choice: 3
 Queue elements: 2 4
 1. Insert
 2. Delete
 3. Display
 4. Exit
 Enter your choice: 2
 Element deleted from queue is: 2
 1. Insert
 2. Delete
 3. Display
```

```
4. Exit
Enter your choice: 4
```

**RESULT :**

|  |  |
|---|---|
| EX.NO: 5 | **QUICK SORT** |
| DATE: |  |

**AIM:**

**ALGORITHM:**

**PROGRAM:**

```
#include<iostream>
 using namespace std;
 int part(int low, int high, int *a) {
 int i, h = high, l = low, p, t;
 p = a[low]; // Pivot element

 while (low < high) {
 while (a[l] < p) {
l++;
}
 while (a[h] > p) {
 h--;
 }
 if (l < h) {
 t = a[l];
 a[l] = a[h];
 a[h] = t;
 } else {
 t = p;
 p = a[l];
 a[l] = t;
 break;
 }
 }
 return h;
 }

void quick(int l, int h, int *a) {
 int index, i;
 if (l < h) {
 index = part(l, h, a);
 quick(l, index - 1, a);
 quick(index + 1, h, a);
 }
 }

int main() {
 int a[100], n, l, h, i;

 cout << "\nEnter the number of elements: ";
cin >> n;

cout << "\nEnter the elements: ";
 for (i = 0; i < n; i++)
cin >> a[i];

 cout << "\nInitial Array: \n";
 for (i = 0; i < n; i++)
 cout << a[i] << "\t";

 h = n - 1;
 l = 0;
 quick(l, h, a);

 cout << "\nAfter Sorting: \n";
 for (i = 0; i < n; i++)
 cout << a[i] << "\t";
 return 0;
 }
```

**OUTPUT:**

```
Enter the number of elements: 5
 Enter the elements: 23
 25
 64
 22
 1

 Initial Array:
 23  25  64  22  1

 After Sorting:
 1  22  23  25  64
```

**RESULT :**

|  |  |
|---|---|
| EX.NO: 6 | **ASCENDING ORDER USING HEAP SORT** |
| DATE: |  |

**AIM:**

**ALGORITHM:**

**PROGRAM :**

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

 void heapify(int arr[], int n, int i) {
 int largest = i;
int l = 2 * i + 1;
 int r = 2 * i + 2;

 if (l < n && arr[l] > arr[largest])
largest = l;
 if (r < n && arr[r] > arr[largest])
 largest = r;
 if (largest != i) {
 swap(arr[i], arr[largest]);
 heapify(arr, n, largest);
 }
 }
 void heapsort(int arr[], int n) {
 for (int i = n / 2 - 1; i >= 0; i--)
 heapify(arr, n, i);
 for (int i = n - 1; i >= 0; i--) {
 // Move current root to the end
 swap(arr[0], arr[i]);

 heapify(arr, i, 0);
 }
 }
 void printarray(int arr[], int n) {
for (int i = 0; i < n; i++)
 cout << arr[i] << " ";
cout << "\n";
 }
 int main() {
 int arr[30], n;
 cout << "\nEnter the number of elements: ";
 cin >> n;
cout << "\nEnter the elements: ";
 for (int i = 0; i < n; i++)
 cin >> arr[i];

cout << "\nInitial Array: \n";
 printarray(arr, n);
 heapsort(arr, n);
 cout << "\nSorted array is: \n";
 printarray(arr, n);
 return 0;
 }
```

**OUTPUT:**

```
Enter the number of elements: 5
Enter the elements: 66
 52
 59
 77
 21

 Initial Array:
 66  52  59  77  21

 Sorted array is:
 21  52  59  66  77
```

**RESULT :**

| | |
|---|---|
| EX.NO: 7 | **KNAPSACK PROBLEM USING GREEDY METHOD** |
| DATE: | |

## AIM:

## ALGORITHM:

**PROGRAM:**

```
#include<iostream>
 #include<vector>
 #include<algorithm>
 using namespace std;
 int knapsack(int capacity, const vector& weights, const vector& profits, int n) {
vector<vector > dp(n + 1, vector(capacity + 1, 0)); // Space between the `>` symbols
 for (int i = 1; i <= n; ++i) {
 for (int w = 1; w <= capacity; ++w) {
if (weights[i - 1] <= w) {
 dp[i][w] = max(profits[i - 1] + dp[i - 1][w - weights[i - 1]], dp[i - 1][w]);
 } else {
dp[i][w] = dp[i - 1][w];
}
 }
 }
 return dp[n][capacity]; // Return the maximum profit stored in dp[n][capacity]
}
int main() {
 int numItems, capacity;
 cout << "Enter the number of items: ";
 cin >> numItems;
 cout << "Enter the capacity of the knapsack: ";
 cin >> capacity;
 vector weights(numItems);
 vector profits(numItems);
 cout << "Enter the weights of the items:\n";
for (int i = 0; i < numItems; ++i) {
cin >> weights[i];
 }
 cout << "Enter the profits of the items:\n";
 for (int i = 0; i < numItems; ++i) {
 cin >> profits[i];
 }
int maxProfit = knapsack(capacity, weights, profits, numItems);
 cout << "The maximum profit is: " << maxProfit << endl;

 return 0;
 }
```

**OUTPUT :**

```
Enter the number of items: 3
Enter the capacity of the knapsack: 3
Enter the weights of the items:
1
 3
 6
 Enter the profits of the items:
2
4
7

 The maximum profit is: 4
```

**RESULT :**

|  |  |
|---|---|
| EX.NO: 8 | **DIVIDE AND CONQUER STRATEGY** |
| DATE: |  |

**AIM:**

**ALGORITHM:**

**PROGRAM :**

```
#include<iostream>
 using namespace std;
 int main() {
 int n, a[30], i, top, mid, bottom, item;
 cout << "Enter how many elements you want: ";
 cin >> n;
cout << "\nEnter the " << n << " elements in ascending order: ";
 for (i = 0; i < n; i++) {
 cin >> a[i];
}
cout << "\nEnter the item to search: ";
 cin >> item;
 bottom = 0;
top = n - 1;
 bool found = false;
 while (bottom <= top) {
 mid = (bottom + top) / 2;
if (item == a[mid]) {
 found = true;
 break;
} else if (item < a[mid]) {
top = mid - 1;
} else {
 bottom = mid + 1;
}
 }
 if (found) {
 cout << "\nBinary search successful";
 cout << "\n" << item << " found at position " << mid + 1 << endl; // Outputting position as 1-based
index
 } else {
 cout << "\nSearch failed, not found" << endl;
 }
 return 0;
 }
```

**OUTPUT:**

```
Enter how many elements you want: 5
Enter the 5 elements in ascending order: 11
 12
 13
 14
 15
Enter the item to search: 14
 Binary search successful
 14 found at position 4
```

**RESULT:**

|  |  |
|---|---|
| EX.NO: 9 | **8- QUEENS ON AN 8X8 MATRICES** |
| DATE: |  |

**AIM:**

**ALGORITHM:**

**PROGRAM :**

```
#include<iostream>
 #include<cmath>
 using namespace std;
 char a[10][10];
int n;
void printmatrix() {
cout << "\n";
for (int i = 0; i < n; i++) {
for (int j = 0; j < n; j++)
cout << a[i][j] << " ";
cout << "\n";
 }
 }

 int getmarkedcol(int row) {
 for (int i = 0; i < n; i++) {
 if (a[row][i] == 'Q')
 return i;
 }
 return -1;
 }
 int feasible(int row, int col) {
 for (int i = 0; i < row; i++) {
 int tcol = getmarkedcol(i);
 if (col == tcol || abs(row - i) == abs(col - tcol))
 return 0;
}
 return 1;
 }
 void nqueen(int row) {
 if (row < n) {
 for (int i = 0; i < n; i++) {
 if (feasible(row, i)) {
```

```
 a[row][i] = 'Q';
 nqueen(row + 1);
 a[row][i] = '.';
 }
 }
 } else {
cout << "\nSolution:\n";
printmatrix();
 }
 }
 int main() {
 cout << "Enter the number of queens: ";
cin >> n;
 for (int i = 0; i < n; i++) {
 for (int j = 0; j < n; j++) {
a[i][j] = '.';
 }
 }
 nqueen(0);
return 0;
 }
```

**OUTPUT :**

```
Enter the number of queens: 4
Solution:
 . Q . .
 . . . Q
 Q . . .
 . . Q .
 Solution:
 . . Q .
 Q . . .
 . . . Q
. Q . .
```

**RESULT :**

|  |  |
|---|---|
| EX.NO: 10 | **VIRTUAL FUNCTION** |
| DATE: |  |

**AIM:**

**ALGORITHM:**

**PROGRAM :**

```
#include<iostream.h>
 #include<math.h>
 using namespace std;
 class Shape {
 public:
 virtual float cal_area() {
 return 0;
 }
 virtual float cal_perimeter() {
 return 0;
 }
 };
 class Sphere : public Shape {
float radius, area, perimeter;
 public:
 void get() {
cout << "\n Enter the radius of the sphere: ";
cin >> radius;
}
float cal_area() {
area = 4 * M_PI * radius * radius;
return area;
 }
 float cal_perimeter() {
 perimeter = 2 * M_PI * radius;
 return perimeter;
 }
};
 int main() {
Shape* shapePtr;
 Sphere sphere;
 shapePtr = &sphere;
sphere.get();
 cout << "\n\tSurface area of the sphere: " << shapePtr->cal_area();
 cout << "\n\tPerimeter (Circumference) of the sphere: " << shapePtr->cal_perimeter();
 return 0;
 }
```

**OUTPUT:**

```
Enter the radius of the sphere: 6
 Surface area of the sphere: 452.389
 Perimeter (Circumference) of the sphere: 37.6991
```

**RESULT:**

| | |
|---|---|
| EX.NO: 11 | **PARAMETERIZED CONSTRUCTOR** |
| DATE: | |

**AIM:**

**ALGORITHM:**

**PROGRAM:**

```
#include<iostream>
 using namespace std;
 class num {
 private:

 int a, b, c;
public:
 num(int m, int j, int k);
 void show() {
 cout << "\na = " << a << " b = " << b << " c = " << c << endl;
 }
 };
 num::num(int m, int j, int k) {
 a = m;
 b = j;
 c = k;
 }
 int main() {
 system("CLS");
num x(4, 5, 7);
 num y(1, 2, 8);
 x.show();
 y.show();
 return 0;
 }
```

**OUTPUT:**

```
a = 4 b = 5 c = 7
 a = 1 b = 2 c = 8
```

**RESULT:**

|  |  |
|---|---|
| EX.NO: 12 | **FRIEND FUNCTION** |
| DATE: |  |

**AIM:**

**ALGORITHM:**

**PROGRAM:**

```
#include<iostream>
 using namespace std;
 class A;
 class B {
private:
 int a;
 float b;
 public:
 friend void display(A, B);
 void get() {
 cout << "\nEnter the integer number: ";
 cin >> a;
 cout << "\nEnter the float number: ";
 cin >> b;
}
 };
 class A {
private:
 int c;
 float d;
 public:
 friend void display(A, B);

 void get() {
 cout << "\nEnter the integer number: ";
 cin >> c;
cout << "\nEnter the float number: ";

cin >> d;
}
 };
 void display(A m, B n) {
 cout << "\nInteger results are: " << m.c << " and " << n.a;
 cout << "\nFloat results are: " << m.d << " and " << n.b;
 }
 int main() {
A x1;
 B x2
 x1.get();
 x2.get();
 display(x1, x2);
 return 0;
 }
```

**OUTPUT:**

```
Enter the integer number: 5
Enter the float number: 2.2
Enter the integer number: 8
 Enter the float number: 7.7

 Integer results are: 5 and 8
 Float results are: 2.2 and 7.7
```

**RESULT:**

|  |  |
|---|---|
| EX.NO: 13 | **FUNCTION OVERLOADING** |
| DATE: |  |

**AIM:**

**ALGORITHM:**

**PROGRAM:**

```
#include <iostream>
#define PI 3.142857142857142857
 int calcarea(int length, int breadth) {
 return length * breadth;
 }
 float calcarea(double base, double height) {
return 0.5 * base * height;
 float calcarea(float radius) {
 return (4.0 / 3.0) * PI * radius * radius * radius;
}
 int main() {
int areal;
 float area2, area3;
 areal = calcarea(10, 20);
 area2 = calcarea(4.5, 2.1);
 area3 = calcarea(3.12145f);
 std::cout << "Area of rectangle is: " << areal << std::endl;
std::cout << "Area of triangle is: " << area2 << std::endl;
 std::cout << "Volume of sphere is: " << area3 << std::endl;
return 0;
 }
```

**OUTPUT:**

```
Area of rectangle is: 200
Area of triangle is: 4.725
 Volume of sphere is: 127.448
```

**RESULT:**

|  |  |
|---|---|
| EX.NO: 14 | **SINGLE INHERITANCE** |
| DATE: |  |

**AIM:**

**ALGORITHM:**

**PROGRAM:**

```
#include<iostream>
 using namespace std;
 class Animal {
public:
 void eat() {
 cout << "Animal is eating." << endl;
}
 };
 class Dog : public Animal {
 public:
void bark() {
 cout << "Dog is barking." << endl;
 }
 };
 int main() {
 Dog myDog;
 myDog.eat();
myDog.bark();
 return 0;
 }
```

**OUTPUT:**

```
Animal is eating.
Dog is barking.
```

**RESULT:**

|  |  |
|---|---|
| EX.NO: 15 | **EMPLOYEE DETAILS USING FILES** |
| DATE: |  |

**AIM:**

**ALGORITHM:**

**PROGRAM:**

```
#include <iostream>
#include <fstream>
#include<string>//  For using std::string
 using namespace std;
 int main() {
 string data; // Using string instead of char array
 ofstream outfile;
outfile.open("emp.txt");
if (!outfile) {
 cerr << "Error opening file for writing!" << endl;
 return 1;
 }
 cout << "Enter the name of employee: ";
 getline(cin, data); // Use getline for strings
 outfile << data << endl;
 cout << "Enter the ID: ";
 getline(cin, data);
 outfile << data << endl;
 cout << "Department: ";
getline(cin, data);
 outfile << data << endl;

cout << "Salary: ";
 getline(cin, data);
 outfile << data << endl;
 outfile.close();
 ifstream infile;

infile.open("emp.txt");
 if (!infile) {
 cerr << "Error opening file for reading!" << endl;
 return 1;
 }
 cout << "\nReading from file:\n";
 while (getline(infile, data)) {
 cout << data << endl;
}
infile.close();
 return 0;
 }
```

**<u>OUTPUT:</u>**

```
Enter the name of employee: Edison
Enter the ID: 1003
 Department: CS
 Salary: 25000

 Reading from file:
 Edison
 1003
 CS
 25000
```

**<u>RESULT:</u>**