Contents

CODE:

```
package Project;
public class Node {
    String name;
    String ID;
    String first_day_of_work;
    String phone_number;
    String address;
    double work_hours;
    double salary;

    Node next;
    Node prev;

    /**
     * Constructor for Node class
     *
     * @param name              the name of the employee
     * @param ID                the ID of the employee
     * @param first_day_of_work the first day of work of the employee
     * @param phone_number      the phone number of the employee
     * @param address           the address of the employee
     * @param work_hours        the work hours of the employee
     * @param salary            the salary of the employee
     */
    public Node(String name, String ID, String first_day_of_work, String
phone_number, String address,
            double work_hours, double salary) {
        this.name = name;
        this.ID = ID;
        this.first_day_of_work = first_day_of_work;
        this.phone_number = phone_number;
        this.address = address;
        this.work_hours = work_hours;
        this.salary = salary;
    }

    /**
     * @return Employee information as a string
     */
    public String toString() {
        return "Name: " + name + " ID: " + ID + " First day of work: " +
first_day_of_work + " Phone number: "
                + phone_number + " Address: " + address + " Work hours: " +
work_hours + " Salary: " + salary;
    }
}
```

```
class EmployeeRecordManagement {
    Node head;
    Node tail;

    EmployeeRecordManagement() {
        head = null;
        tail = null;
    }

    /**
     * Checks if a string is a valid phone number
     *
     * @param phone_number the phone number to check
     * @return true if the phone number is valid, false otherwise
     */
    public boolean isValidPhoneNumber(String phone_number) {
        if (phone_number.length() != 10) {
            return false;
        }
        for (int i = 0; i < phone_number.length(); i++) {
            if (phone_number.charAt(i) < '0' || phone_number.charAt(i) > '9')
{
                return false;
            }
        }
        return true;
    }

    /**
     * Inserts an employee into the list
     *
     * @param name              the name of the employee
     * @param ID                the ID of the employee
     * @param first_day_of_work the first day of work of the employee
     * @param phone_number      the phone number of the employee
     * @param address           the address of the employee
     * @param work_hours        the work hours of the employee
     * @param salary            the salary of the employee
     */
    public void insertEmployee(String name, String ID, String
first_day_of_work, String phone_number, String address,
            double work_hours, double salary) {
        // Check if phone number is valid
        if (!isValidPhoneNumber(phone_number)) {
            System.out.println("Invalid phone number!");
            return;
        }
        // Create a new node
        Node newNode = new Node(name, ID, first_day_of_work, phone_number,
address, work_hours, salary);
        // Check if list is empty
        if (head == null) {
            head = newNode;
            tail = newNode;
            System.out.println("Employee added successfully!");
            return;
```

```
        }
        // Check if employee already exists
        Node temp = head;
        while (temp != null) {
            if (temp.ID.equals(ID)) {
                System.out.println("Employee already exists!");
                return;
            }
            temp = temp.next;
        }
        // Insert employee
        temp = head;
        while (temp != null) {
            if (Integer.parseInt(temp.ID) > Integer.parseInt(ID)) {
                if (temp == head) {
                    newNode.next = head;
                    head.prev = newNode;
                    head = newNode;
                } else {
                    newNode.next = temp;
                    newNode.prev = temp.prev;
                    temp.prev.next = newNode;
                    temp.prev = newNode;
                }
                System.out.println("Employee added successfully!");
                return;
            }
            temp = temp.next;
        }
        tail.next = newNode;
        newNode.prev = tail;
        tail = newNode;
        System.out.println("Employee added successfully!");
    }

    /**
     * Deletes an employee from the list
     *
     * @param ID the ID of the employee to delete
     * @return 0 if Employee deleted successfully and -1 if Employee not
found
     */
    public int deleteEmployee(String ID) {
        Node temp = head;
        while (temp != null) {
            if (temp.ID.equals(ID)) {
                if (temp == head) {
                    head = head.next;
                    head.prev = null;
                } else if (temp == tail) {
                    tail = tail.prev;
                    tail.next = null;
                } else {
                    temp.prev.next = temp.next;
                    temp.next.prev = temp.prev;
                }
                System.out.println("Employee deleted successfully!");
```

```
                return 0;
            }
            temp = temp.next;
        }
        System.out.println("Employee not found!");
        return -1;
    }

    /**
     * Searches for an employee in the list
     *
     * @param ID the ID of the employee to search for
     */
    public void searchEmployee(String ID) {
        Node temp = head;
        while (temp != null) {
            if (temp.ID.equals(ID)) {
                System.out.println("Employee found!");
                return;
            }
            temp = temp.next;
        }
        System.out.println("Employee not found!");
    }

    /**
     * Prints employee records with matching ID
     *
     * @param ID the ID of the employee to print
     */
    public void printEmployee(String ID) {
        Node temp = head;
        while (temp != null) {
            if (temp.ID.equals(ID)) {
                System.out.println(temp);
                return;
            }
            temp = temp.next;
        }
        System.out.println("Employee not found!");
    }

    /**
     * Updates an employee record
     *
     * @param ID                the ID of the employee to print
     * @param name              the name of the employee to print
     * @param first_day_of_work the first day of work of the employee to
print
     * @param phone_number      the phone number of the employee to print
     * @param address           the address of the employee to print
     * @param work_hours        the work hours of the employee to print
     * @param salary            the salary of the employee to print
     */
    public void updateEmployeeRecord(String ID, String name, String
first_day_of_work, String phone_number,
            String address, double work_hours, double salary) {
```

```
        Node temp = head;
        while (temp != null) {
            if (temp.ID.equals(ID)) {
                temp.name = name;
                temp.first_day_of_work = first_day_of_work;
                temp.phone_number = phone_number;
                temp.address = address;
                temp.work_hours = work_hours;
                temp.salary = salary;
                System.out.println("Employee updated successfully!");
                return;
            }
            temp = temp.next;
        }
        System.out.println("Employee not found!");
    }

    /**
     * Updates the salary of an employee update the salary of an employee
where if
     * the working hours is more than 32 h
     * then the employee gets 2% additional of the salary for each hour
     *
     * @param ID the ID of the employee to update
     */
    public void updateSalary(String ID) {
        Node temp = head;
        while (temp != null) {
            if (temp.ID.equals(ID)) {
                if (temp.work_hours > 32) {
                    temp.salary += (temp.work_hours - 32) * 0.02 *
temp.salary;
                }
                System.out.println("Salary updated successfully!");
                return;
            }
            temp = temp.next;
        }
        System.out.println("Employee not found!");
    }

    /**
     * Prints all employees
     */
    public void printAllEmployees() {
        Node temp = head;
        while (temp != null) {
            System.out.println(temp);
            temp = temp.next;
        }
    }

    public static void main(String[] args) {
        // Testing all the methods
        EmployeeRecordManagement employeeRecordManagement = new
EmployeeRecordManagement();
        employeeRecordManagement.insertEmployee("Ahmed", "1", "Saturday",
```

```
"0512345678", "Jeddah", 40, 4321.2);
        employeeRecordManagement.insertEmployee("Mohamed", "2", "Sunday",
"0512635678", "Jeddah", 32, 21.2);
        employeeRecordManagement.insertEmployee("Ali", "3", "Tuesday",
"0535397178", "Jeddah", 10, 411.2);
        employeeRecordManagement.insertEmployee("Khaled", "4", "Friday",
"0598945678", "Jeddah", 44, 321.12);
        employeeRecordManagement.insertEmployee("Sayed", "5", "Monday",
"1234567890", "Jeddah", 23, 1050.2);
        employeeRecordManagement.insertEmployee("Salem", "6", "Wednessday",
"1234567890", "Jeddah", 16, 21.2);
        employeeRecordManagement.insertEmployee("Hossam", "6", "Thursday",
"1234567890", "Jeddah", 38, 4321.2);

        System.out.println("-------------------------");
        System.out.println(employeeRecordManagement.deleteEmployee("3"));
        System.out.println(employeeRecordManagement.deleteEmployee("99"));
        employeeRecordManagement.searchEmployee("4");
        employeeRecordManagement.printEmployee("6");
        employeeRecordManagement.updateEmployeeRecord("6", "Ameen",
"Thursday", "01111110000", "Jeddah", 34, 10000);
        employeeRecordManagement.printEmployee("6");
        employeeRecordManagement.updateSalary("6");
        employeeRecordManagement.printEmployee("6");
        employeeRecordManagement.printAllEmployees();
    }
}
```

OUTPUT: