Institute of Computer Technology

B. Tech. Computer Science and Engineering

Sub: ESFP – II

Course Code: 2CSE203

Practical – 8

Name: Jaymin Gondaliya

Enrollment No: 23162171007

Sem - 2

Branch: CS

Class: B

Batch: 25

Objective:

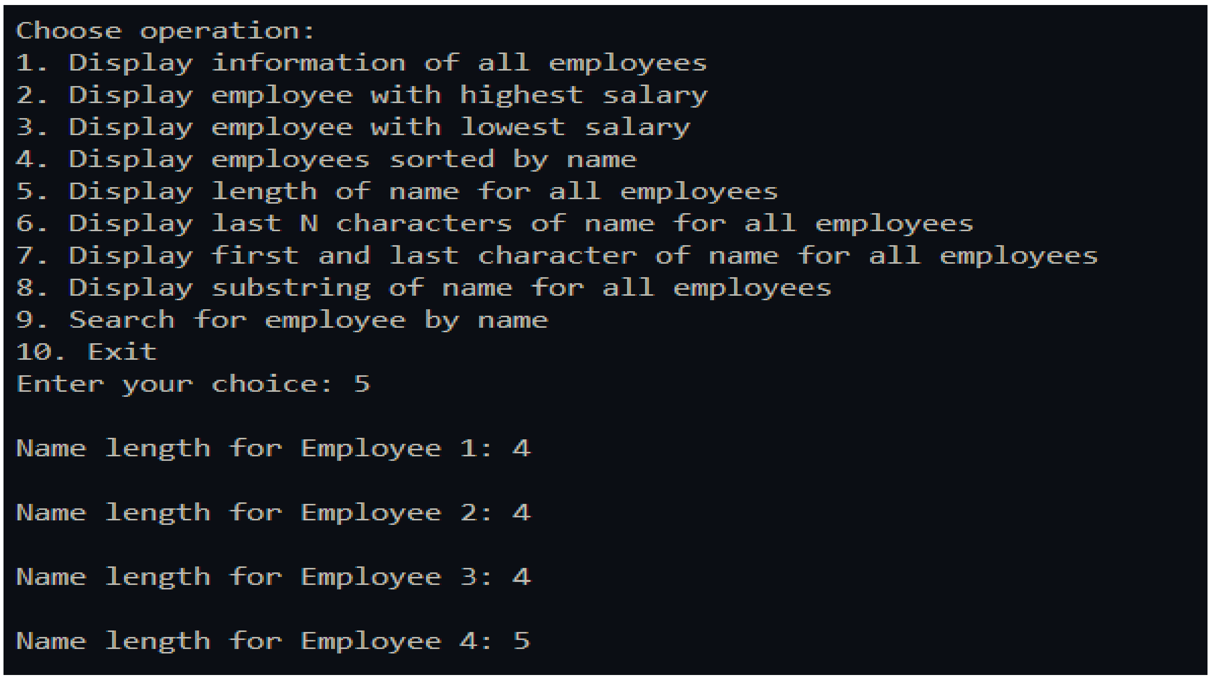
To implement string functions.

Problem Definition-1: Complete the code for the object assigned to you to satisfy following specifications.

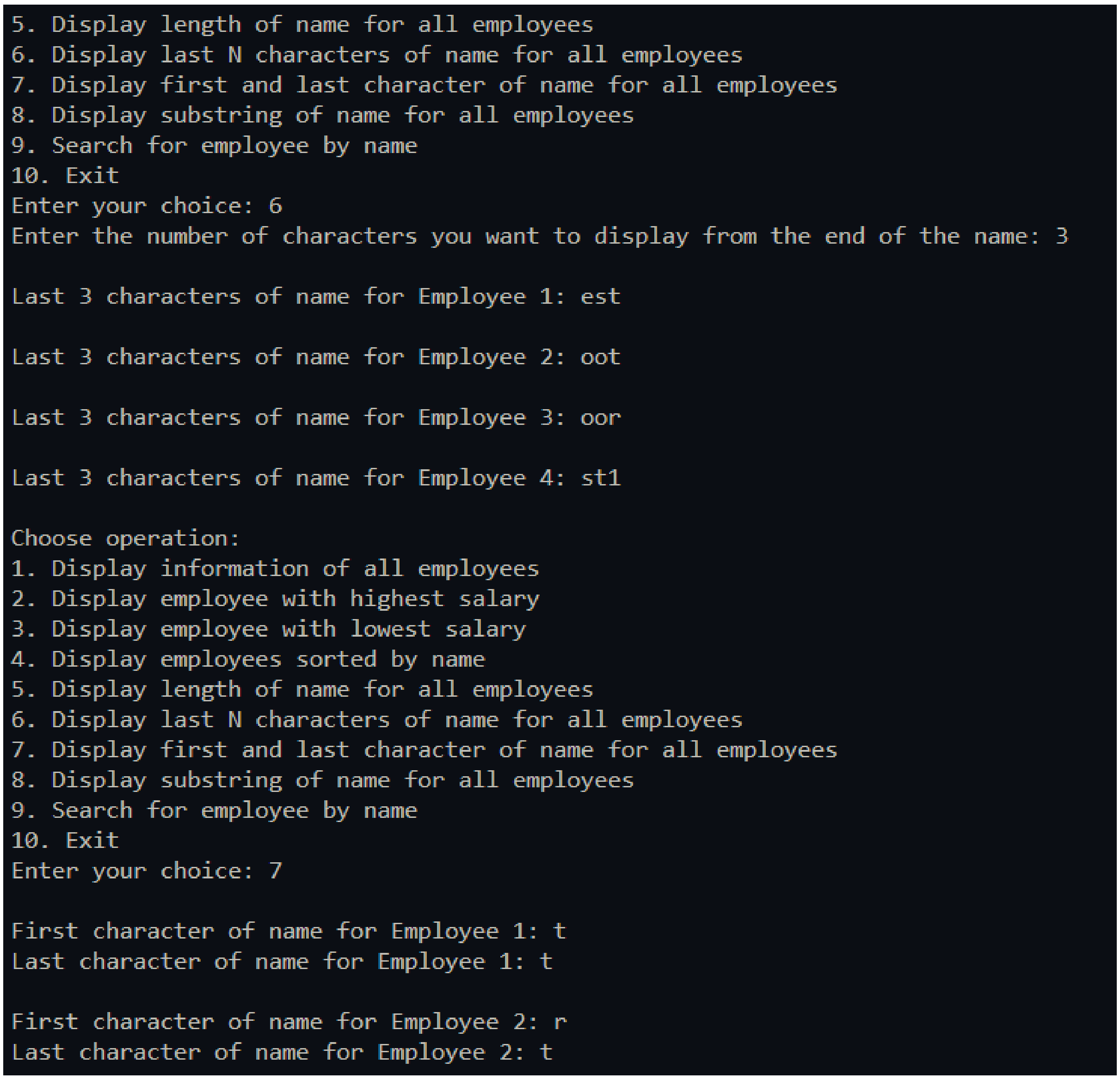
Code:

```
#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

const int MAX_EMPLOYEES = 100;

class Employee
{
private:
    string employeeID;
    string name;
    int age;
    string designation;
    double salary;

public:
    Employee()
    {
        employeeID = "";
        name = "";
        age = 0;
        designation = "";
        salary = 0.0;
    }

    Employee(string id, string n, int a, string d, double s)
    {
        employeeID = id;
        name = n;
        age = a;
        designation = d;
        salary = s;
    }

    double getSalary()
    {
        return salary;
    }

    void displayInfo()
    {
        cout << "-------------------------------------------------------------
--------" << endl;
        cout << "| Employee ID |      Name      | Age
|   Designation   |    Salary    |" << endl;
        cout << "-------------------------------------------------------------
--------" << endl;
        cout << "| " << setw(12) << employeeID << " | " << setw(14) << name <<
" | " << setw(3) << age << " | " << setw(15) << designation << " | " <<
setw(10) << salary << " |" << endl;
        cout << "-------------------------------------------------------------
--------" << endl;
    }

    int getAge()
    {
        return age;
    }

    string getName()
    {
        return name;
    }

    string getDesignation()
    {
        return designation;
    }

    string getEmployeeID()
    {
        return employeeID;
    }

    string getLastNCharacters(int n)
    {
        return name.substr(name.length() - n);
    }

    char getFirstCharacter()
    {
        return name[0];
    }

    char getLastCharacter()
    {
        return name[name.length() - 1];
    }

    string getSubstring(int start, int length)
    {
        return name.substr(start, length);
    }
};

int main()
{
    int numEmployees;

    cout << "How many employee information do you want to store? ";
    cin >> numEmployees;

    if (numEmployees > MAX_EMPLOYEES)
    {
        cout << "Number of employees exceeds maximum capacity." << endl;
        return 1;
    }

    Employee employees[MAX_EMPLOYEES];
    string id, name, designation;
    int age;
    double salary;
    for (int i = 0; i < numEmployees; ++i)
    {
        cout << "\nEnter information for Employee " << i + 1 << endl;
        cout << "Enter Employee ID: ";
        cin >> id;
        cout << "Enter Name: ";
        cin.ignore();
        getline(cin, name);
        cout << "Enter Age: ";
        cin >> age;
        cout << "Enter Designation: ";
        cin.ignore();
        getline(cin, designation);
        cout << "Enter Salary: ";
        cin >> salary;

        employees[i] = Employee(id, name, age, designation, salary);
    }

    int choice;
    string searchName;
    bool found;
    do
    {
        cout << "\nChoose operation: " << endl;
        cout << "1. Display information of all employees" << endl;
        cout << "2. Display employee with highest salary" << endl;
        cout << "3. Display employee with lowest salary" << endl;
        cout << "4. Display employees sorted by name" << endl;
        cout << "5. Display length of name for all employees" << endl;
        cout << "6. Display last N characters of name for all employees" <<
endl;
        cout << "7. Display first and last character of name for all
employees" << endl;
        cout << "8. Display substring of name for all employees" << endl;
        cout << "9. Search for employee by name" << endl;
        cout << "10. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice)
        {
        case 1:
            for (int i = 0; i < numEmployees; ++i)
            {
                cout << "\nEmployee " << i + 1 << ":" << endl;
                employees[i].displayInfo();
            }
            break;
        case 2:
        {
            int highestSalaryIndex = 0;
            double highestSalary = employees[0].getSalary();
            for (int i = 1; i < numEmployees; ++i)
            {
                if (employees[i].getSalary() > highestSalary)
                {
                    highestSalary = employees[i].getSalary();
                    highestSalaryIndex = i;
                }
            }
            cout << "Employee with highest salary: " << endl;
            employees[highestSalaryIndex].displayInfo();
            break;
        }
        case 3:
        {
            int lowestSalaryIndex = 0;
            double lowestSalary = employees[0].getSalary();
            for (int i = 1; i < numEmployees; ++i)
            {
                if (employees[i].getSalary() < lowestSalary)
                {
                    lowestSalary = employees[i].getSalary();
                    lowestSalaryIndex = i;
                }
            }
            cout << "Employee with lowest salary: " << endl;
            employees[lowestSalaryIndex].displayInfo();
            break;
        }
        case 4:
            for (int i = 0; i < numEmployees - 1; ++i)
            {
                for (int j = 0; j < numEmployees - i - 1; ++j)
                {
                    if (employees[j].getName() > employees[j + 1].getName())
                    {
                        Employee temp = employees[j];
                        employees[j] = employees[j + 1];
                        employees[j + 1] = temp;
                    }
                }
            }
            cout << "\nEmployees sorted by name:" << endl;
            for (int i = 0; i < numEmployees; ++i)
            {
                cout << "\nEmployee " << i + 1 << ":" << endl;
                employees[i].displayInfo();
            }
            break;

        case 5:
            for (int i = 0; i < numEmployees; ++i)
            {
                cout << "\nName length for Employee " << i + 1 << ": " <<
employees[i].getName().length() << endl;
            }
            break;
        case 6:
            int n;
            cout << "Enter the number of characters you want to display from
the end of the name: ";
            cin >> n;
            for (int i = 0; i < numEmployees; ++i)
            {
                cout << "\nLast " << n << " characters of name for Employee "
<< i + 1 << ": " << employees[i].getLastNCharacters(n) << endl;
            }
            break;
        case 7:
            for (int i = 0; i < numEmployees; ++i)
            {
                cout << "\nFirst character of name for Employee " << i + 1 <<
": " << employees[i].getFirstCharacter() << endl;
                cout << "Last character of name for Employee " << i + 1 << ":
" << employees[i].getLastCharacter() << endl;
            }
            break;
        case 8:
            int start, length;
            cout << "Enter starting position for substring: ";
            cin >> start;
            cout << "Enter length of substring: ";
            cin >> length;
            for (int i = 0; i < numEmployees; ++i)
            {
                cout << "\nSubstring of name for Employee " << i + 1 << ": "
<< employees[i].getSubstring(start, length) << endl;
            }
            break;
        case 9:
            cout << "Enter the name you want to search for: ";
            cin.ignore();
            getline(cin, searchName);
            found = false;
            for (int i = 0; i < numEmployees; ++i)
            {
                if (employees[i].getName() == searchName)
                {
                    cout << "\nEmployee found:" << endl;
                    employees[i].displayInfo();
                    found = true;
                    break;
                }
            }
            if (!found)
            {
                cout << "\nEmployee not found." << endl;
            }
            break;
        case 10:
            cout << "Exiting program." << endl;
            break;
        default:
            cout << "Invalid choice. Please enter again." << endl;
            break;
        }
    } while (choice != 10);



    return 0;
}
```

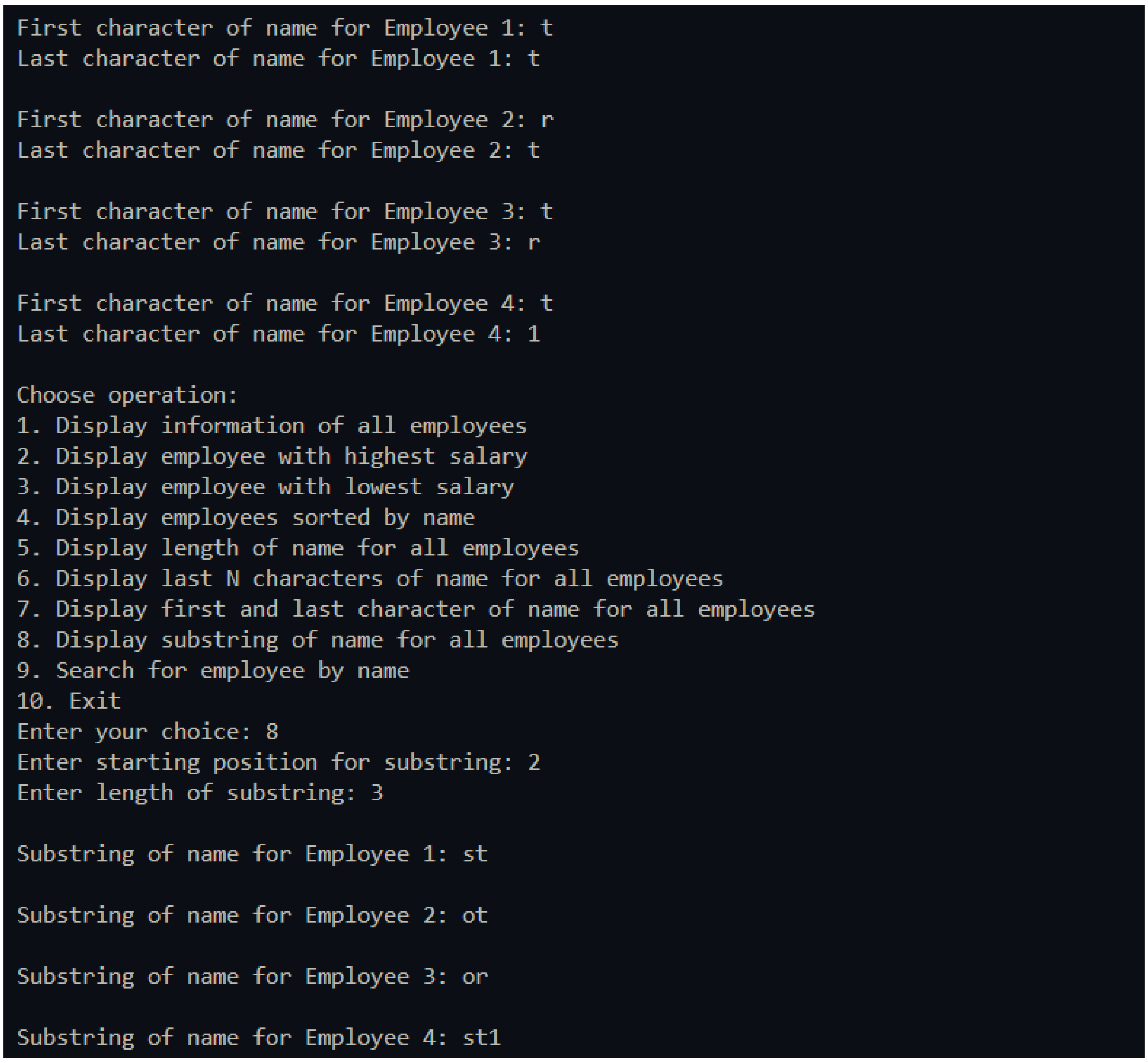
Output:



```
Choose operation:
1. Display information of all employees
2. Display employee with highest salary
3. Display employee with lowest salary
4. Display employees sorted by name
5. Display length of name for all employees
6. Display last N characters of name for all employees
7. Display first and last character of name for all employees
8. Display substring of name for all employees
9. Search for employee by name
10. Exit
Enter your choice: 5

Name length for Employee 1: 4

Name length for Employee 2: 4

Name length for Employee 3: 4

Name length for Employee 4: 5
```



```
5. Display length of name for all employees
6. Display last N characters of name for all employees
7. Display first and last character of name for all employees
8. Display substring of name for all employees
9. Search for employee by name
10. Exit
Enter your choice: 6
Enter the number of characters you want to display from the end of the name: 3

Last 3 characters of name for Employee 1: est

Last 3 characters of name for Employee 2: oot

Last 3 characters of name for Employee 3: oor

Last 3 characters of name for Employee 4: st1

Choose operation:
1. Display information of all employees
2. Display employee with highest salary
3. Display employee with lowest salary
4. Display employees sorted by name
5. Display length of name for all employees
6. Display last N characters of name for all employees
7. Display first and last character of name for all employees
8. Display substring of name for all employees
9. Search for employee by name
10. Exit
Enter your choice: 7

First character of name for Employee 1: t
Last character of name for Employee 1: t

First character of name for Employee 2: r
Last character of name for Employee 2: t
```



```
First character of name for Employee 1: t
Last character of name for Employee 1: t

First character of name for Employee 2: r
Last character of name for Employee 2: t

First character of name for Employee 3: t
Last character of name for Employee 3: r

First character of name for Employee 4: t
Last character of name for Employee 4: 1

Choose operation:
1. Display information of all employees
2. Display employee with highest salary
3. Display employee with lowest salary
4. Display employees sorted by name
5. Display length of name for all employees
6. Display last N characters of name for all employees
7. Display first and last character of name for all employees
8. Display substring of name for all employees
9. Search for employee by name
10. Exit
Enter your choice: 8
Enter starting position for substring: 2
Enter length of substring: 3

Substring of name for Employee 1: st

Substring of name for Employee 2: ot

Substring of name for Employee 3: or

Substring of name for Employee 4: st1
```